ESCUELA POLITÉCNICA SUPERIOR
UNIVERSIDAD CARLOS III DE MADRID

INGENIERÍA SUPERIOR INFORMÁTICA

“Creación de un mashup con Selenium”

Autor:

Miguel Javier de la Rosa Escolante

Tutor:

Vicente Luque Centeno

Fecha:

Mayo 2011

A mis padres José y Mary Carmen, cuyos valores me han inculcado, a mi hermana María, que me ha apoyado siempre incondicionalmente y también a mi tío Javier que junto con mis padres es uno de los artificies de que yo haya realizado estos estudios. Mención especial a mis abuelos, ya que me hubiera gustado que todos hubieran vivido para verme terminar esta carrera.

También quería agradecer a todas aquellas personas que están o han estado a mi lado alguna vez, y que tanto me han soportado durante los momentos duros del transcurso de mi periodo universitario, así como tanto se han alegrado en los buenos.

Muchas gracias a todos.

Índice

Índice de figuras

Índice de tablas

1. Introducción

En esta memoria se van a exponer todos los procedimientos y detalles de la implementación de este proyecto que lleva por título: *"Creación de un mashup con Selenium*".

Para comprender de qué trata el proyecto, vamos a explicar primero el significado de cada una de las palabras que componen su título.

Un mashup, es una aplicación que integra, de una manera eficiente y sencilla, diferentes datos de distintas fuentes para producir resultados en un formato homogéneo, con el objetivo de que estos, puedan resultar más productivos para los usuarios que los datos originales mostrados por cada una de las fuentes.

El término Selenium, hace referencia a la tecnología a utilizar para desarrollar el proyecto. Esta tecnología es un conjunto de herramientas cuya funcionalidad es la de ayudar al desarrollo de pruebas automáticas para aplicaciones basadas en Web. En este proyecto, esta tecnología será utilizada para navegar por los distintos elementos de las webs seleccionadas como fuente de datos. Todos los aspectos relacionados con esta tecnología serán explicados en detalle en el apartado 2.2.

Una vez definido los términos contenidos en el título del presente proyecto, podemos definir que el objetivo del mismo es la creación de una aplicación (mashup) que integre el contenido resultante de las búsquedas en distintos portales webs de venta de productos tecnológicos y para ello nos serviremos de una tecnología destinada al desarrollo de pruebas automáticas para aplicaciones web, Selenium.

El aspecto de tener que utilizar una tecnología determinada y específica que no está especialmente realizada para el fin del proyecto, la tecnología Selenium, añade dificultad al desarrollo del proyecto. Al no ser la tecnología más apropiada para la funcionalidad de nuestra aplicación, su uso ha presentado algunos inconvenientes en el aspecto de eficiencia, pero a la vez ha supuesto un reto el tener que utilizar una tecnología específica.

En este capítulo de esta memoria, vamos a ver los apartados que engloban: la motivación del presente proyecto, la temática elegida en cuanto a los portales, los objetivos del proyecto y el contenido de la memoria, donde explicaremos como se irán exponiendo las bases teóricas del mismo en el presente documento.

1.1. Motivación

El auge del comercio electrónico global, en especial el comercio de productos electrónicos, ha propiciado la aparición de numerosos portales donde se ofrecen este tipo de productos.

El mismo producto se ofrece en diferentes sitios con sede en diferentes lugares, lo que propicia que los precios de estos varíen dependiendo de la empresa, la localización de esta o el cambio de moneda extranjera, entre otros muchos factores.

Esta diversidad de portales, de características y de precios, conlleva que el usuario que quiera realizar una compra se encuentre con una multitud de dificultades para realizar una búsqueda efectiva y encontrar la mejor oferta. Así, sin un buscador que agrupe todos estos portales, el usuario tiene que dedicar mucho tiempo a las tareas de buscar que tiendas son las más importantes primero y luego a la búsqueda una por una de los productos que requiera. También tendrá que seleccionar los resultados de cada una de ellas para poder comparar y elegir la oferta que mejor le conviene.

Con la aplicación de este proyecto, se ayuda a minimizar estos inconvenientes, ofreciendo una solución mejor que la búsqueda manual que los usuarios tienen que realizar para conseguir esta selección y comparación de resultados. La idea de aglutinar la información relevante de diferentes portales en una aplicación, cobra una importancia vital en el sentido de facilitar la búsqueda de información a los usuarios.

En cuanto a los mashups, hoy en día se están extendiendo este tipo de aplicaciones debido a su utilidad práctica. Existen de diversas temáticas, como por ejemplo mashups de noticias, de productos, de vuelos, de ofertas de seguros, de hoteles…

Es muy interesante utilizar este tipo de aplicaciones, y cada vez son más solicitadas por los usuarios, así año a año se va incrementado su uso, hasta tal punto que los propios portales modifican su contenido para ajustarle a las características adecuadas para aparecer como información principal en los mashups más importantes.

Los mashups son aplicaciones capaces de procesar el código HTML de las webs que utilizan como fuente de información, y procesarlo. De tal manera que el mashup dialogue como si se tratara de un usuario con el portal web y pueda rellenar formularios de manera automática en las webs, realizando búsquedas en ellas y obtener la información devuelta por las mismas, para su posterior procesamiento.

El proceso de desarrollo de un proyecto de estas características comprende el análisis de las fuentes de información de las que se quiera extraer su contenido, el proceso de extracción de cada una y la síntesis de la misma para adaptarla a una estructura común en la que se pueda recopilar, para ser mostrada de forma conjunta y homogénea en la aplicación.

Para este proceso de extracción, he utilizado la tecnología Selenium, que como hemos dicho en el punto anterior es una tecnología creada para la realización de pruebas automáticas en páginas web, y de la cual hemos extraído su funcionalidad para utilizarla como función de navegación automática y de comunicación con los portales web seleccionados, mediante sus elementos HTML.

Para poder realizar esta extracción de información relevante de las distintas fuentes, se necesita analizar una por una como está estructurada su información en el código HTML. La mayoría de las veces este proceso es complicado, ya que la información de los portales no está bien estructurada para su extracción, si no para mostrarla al público. Así, este proceso de extracción de información se convierte en una tarea más complicada que si nos encontráramos con documentos HTML bien estructurados.

Como conclusión, la motivación principal es la de desarrollar un sistema capaz de integrar la información y contenidos ofrecidos por distintos portales de venta de productos tecnológicos y que actúe como un único portal que recopile la información de todos ellos.

Así, resumiendo, los distintos puntos que sirven de motivación para este proyecto son:

- El estudio y desarrollo de aplicaciones capaces de interactuar con sitios web de manera inteligente.

- La integración de información no estructurada procedente de estos distintos sitios web y su transformación en una representación bien estructurada y homogénea.

- Mostrar en una misma aplicación una serie de productos obtenidos de diferentes fuentes, posibilitando así al usuario la consulta de todos ellos en un solo lugar, comparar sus informaciones y tener una visión homogénea de sus contenidos de manera sencilla y rápida.

1.2. Objetivos

A continuación se describen detalladamente, cada uno de los objetivos marcados inicialmente para el desarrollo de este proyecto:

- Integrar bajo una misma aplicación el contenido de los portales de venta de productos tecnológicos de los sectores más importantes y utilizados tanto a nivel internacional como a nivel nacional. Con el objetivo de que el usuario pudiera acceder simultáneamente a todos ellos a través de una única interfaz.

- Mostrar la información más relevante de cada producto. Estas son el nombre, el precio y la fotografía como información básica, para que el usuario pueda ver solo las características más importantes y así realizar una comparación de resultados más eficiente.

- Implementar una aplicación capaz de interactuar con el navegador web de manera "inteligente". Que siguiera determinados enlaces, que rellenara determinados formularios y que extrajera la información no estructurada de las páginas web de manera automática.

- El estudio de la tecnología Selenium para llevar a cabo esta navegación automática. Una tecnología que está diseñada para la realización de pruebas automáticas en sitios webs, de la cual queremos obtener su funcionalidad y aplicarla en el desarrollo de la extracción automática de información de distintos portales web.

- Que la interfaz de la aplicación cumpliera con los propósitos de accesibilidad y navegabilidad, para facilitar así el acceso a la misma a los distintos tipos de usuario que puedan beneficiarse de ella.

1.3. Temática

En cuanto a la temática de las fuentes seleccionadas para realizar las búsquedas, se ha elegido el sector de productos tecnológicos. La razón de esta elección radica en que es uno de los sectores actualmente en alza, y cada vez utilizado por un mayor número de usuarios. La temática del mismo, es independiente del desarrollo del proyecto, ya que habiendo elegido otro sector distinto, el desarrollo del proyecto se hubiera producido de una forma muy similar a la actual.

1.3.1. Selección de portales

Para la elección de las fuentes de datos, se ha realizado una selección de los portales de venta de productos tecnológicos basándonos en los siguientes criterios:

- Portales con gran relevancia en el comercio internacional.
- Portales con gran relevancia en el comercio nacional.
- Portales que permitan el envío de productos a España. Esto es debido a que la aplicación se ha diseñado pensando en que los potenciales usuarios de la misma residirán en este país.

Así, expondremos la selección final de portales de venta de productos tecnológicos, ordenándolos por la localización de la empresa:

1.3.1.1. Portales ubicados en Estados Unidos

- **Amazon.com:** es una compañía de comercio electrónico con sede en Seattle (Washington). Es una de las compañías de comercio electrónico más grandes. Actualmente, junto con Ebay, es quien realiza más transacciones de venta de productos online en todo el mundo. Además de productos tecnológicos, ofrece diversas líneas de productos como ropa, libros, material de oficina… No posee sitio web específico para España, pero sí realiza envíos internacionales al país.

- **Ebay.es:** está compañía con sede en San José (California), es otra de las grandes compañías internacionales de comercio electrónico. Originariamente se fundó como una red de subastas por Internet, pero actualmente (aunque es líder indiscutible en este tipo de transacciones) contempla dos tipos de comercio electrónico:

    - **Modalidad Subasta:** es la modalidad estrella del portal, en ella el vendedor fija un precio de salida y un periodo de tiempo para el producto. Durante este periodo los clientes pueden realizar sus pujas. Finalmente una vez concluido el periodo, el cliente que haya realizado la puja más alta adquirirá el producto.
    - **Modalidad "Anuncio clasificado":** al contrario que la modalidad de subastas, el vendedor expone un anuncio en el que fija

un precio para su producto. El cliente que esté dispuesto a pagar ese precio se llevará el producto.

Ebay posee un sitio web específico para España, y al ser un portal donde los vendedores son los usuarios, es posible realizar envíos internaciones y nacionales a España.

1.3.1.2. Portales ubicados en China

- **FocalPrice.com:** es una compañía con sede en Hong Kong (China), que ofrece una gran cantidad de productos tecnológicos a precios muy reducidos. Posee una gran variedad de productos, desde accesorios para Iphone hasta teléfonos móviles, pasando por cámaras, reproductores MP3, juguetes, videojuegos, relojes… Aunque no tiene sede en España, sí permite el envío de productos al país.

- **DealExtreme.com:** compañía muy parecida a FocalPrice.com, también con sede en Hong Kong (China). El tipo de productos que oferta también es muy similar al portal anterior, por lo que contempla una gran variedad. Tampoco tiene sede en España, pero sí permite el envío de productos al país.

1.3.1.3. Empresas con sedes en España.

- **Optize.es:** empresa española con sede en Madrid. Es uno de los portales más utilizados en España a la hora de comprar productos tecnológicos online. Posee una gran variedad de productos ofertados, que engloban las principales categorías de productos tecnológicos. Realiza envíos a España.

- **PC-Online.net:** empresa española con sede principal en Granada. Aunque menos conocida que el resto de las anteriores explicadas, también es utilizada por un gran número de consumidores. Posee un catálogo de productos un poco más limitado que el resto, pero no por ello deja de incluir una gran varie-

dad de categorías. Posee diversas sedes en España y envía productos desde ellas al país.

- **Dynos.es:** empresa española con sede principal en Granada. Al igual que PC-Online, es poco conocida en comparación con el resto de portales contemplados, pero también es bastante utilizada en el ámbito nacional. Su catálogo de productos tampoco es muy abultado pero incluye una gran variedad de categorías, entre las que se encuentran: ordenadores, telefonía, televisiones…. Realiza envíos a España.

Así, como se puede ver, hemos cubierto las empresas más importantes del sector, tanto internacionales como nacionales. Esta elección provoca que el rango de potenciales usuarios de la aplicación de este proyecto sea muy elevado, ya que son sitios muy demandados a la hora de realizar la compra de este tipo de productos. El objetivo era realizar la mayor cobertura posible en cuanto a portales utilizados por los usuarios para este fin.

1.4. Contenido de la memoria

En este apartado explicaremos como se ha organizado el contenido del presente documento, con el objetivo de que el lector se haga una idea de la información que se encontrará en el mismo.

Después de este capítulo de introducción se encuentra el capítulo del estado del arte. Éste contiene una revisión de las bases teóricas de la arquitectura escogida para el proyecto, así como de las tecnologías utilizadas en su desarrollo. En él se realiza una comparación con otras tecnologías que se podían haber utilizado en lugar de las usadas y se dan las razones de por qué se han escogido unas opciones en detrimento de otras.

El tercer capítulo cubre los aspectos de análisis de la aplicación desarrollada. En él se especifica con detalle el problema a resolver con este proyecto. También se especifican los requisitos y casos de uso de la aplicación exponiendo toda la información relevante a cada uno de ellos.

El capítulo cuarto trata todo lo referente al diseño de la aplicación, así se empieza explicando la arquitectura elegida y luego se va explicando capa por capa (nivel por nivel) de la arquitectura detalladamente. De cada capa se explica cómo funciona internamente en la aplicación y los componentes por los que está formada cada una. También se comentará como se ha realizado el diseño de la interfaz.

El capítulo quinto se especifica el funcionamiento del sistema, es decir cómo se comporta el sistema para las distintas funcionalidades. Así diferenciaremos entre la realización de búsquedas (que pueden ser específicas o genéricas) y las operaciones con los resultados, como por ejemplo navegar visualizándolos o realizar alguna ordenación de los mismos.

El capítulo sexto trata sobre los detalles de los aspectos de la implementación de la aplicación. Así lo primero que vemos es el diagrama de clases genéricos para tener una idea visual rápida de cómo se estructura la implementación. Más tarde nos adentramos en la implementación de cada capa. Se verá también las características de la aplicación y el capítulo concluirá explicando cómo se ha utilizado la tecnología Selenium en el desarrollo de la aplicación y como se han gestionado los posibles errores.

En el capítulo séptimo se realizará la evaluación de la aplicación indicando los aspectos positivos y negativos de la misma. Analizaremos cada uno de estos aspectos indicando en que puntos la aplicación tiene fuerza y en qué puntos está un poco más débil.

El capítulo octavo es una comparativa de la realización de búsquedas con la aplicación del proyecto, frente a la búsqueda manual y otras aplicaciones similares.

En el último capítulo se ofrecen las conclusiones personales de haber desarrollado el proyecto, las futuras mejoras que es posible realizar para perfeccionar la funcionalidad de la aplicación y también los datos temporales y económicos que ha supuesto el desarrollo del proyecto.

Finalmente se han decidido incluir dos anexos que se creen de una importancia fundamental. El anexo A es un breve manual de usuario de la aplicación, incluido con el objetivo de poder facilitar al usuario la interacción con la misma. Aunque es una aplicación muy intuitiva y se caracteriza por su sencillez, que permite interactuar fácilmente con ella sin haberse documentado previamente con el manual, se cree oportuno incluirle para facilitar este proceso a los usuarios que no estén familiarizados con este tipo de aplicaciones. En cuanto al Anexo B, se trata de la muestra y explicación de las partes más relevantes del código implementado para el desarrollo de la aplicación. También se encontrará al final del documento la bibliografía utilizada para la elaboración del proyecto así como los enlaces de consulta.

2. Estado del arte

En este capítulo se van a ir exponiendo las diversas tecnologías y estándares utilizados en este proyecto. Al haber sido múltiples elementos los que se han utilizado, se ha decidido dividir cada una de ellas en un apartado distinto para facilitar la estructuración del capítulo.

En cada uno de los apartados se muestra al lector las características principales de cada tecnología, la justificación de por qué se ha elegido y en qué medida se han utilizado en el proyecto. También las razones de su elección comparándolas con tecnologías similares que se podían haber utilizado en su lugar, al igual, incluiremos unas referencias bibliográficas para que el lector pueda ampliar la información aquí recogida.

Primero se ofrecerá una explicación de la arquitectura escogida para desarrollar este proyecto, donde se detallará su funcionalidad junto con un diagrama de la misma. Luego veremos la tecnología más importante y referente de este proyecto, pues es la única de uso obligatorio, que es Selenium, una tecnología desarrollada para realizar pruebas automáticas de webs, como se ha comentado en el capítulo anterior.

A continuación se explicarán las demás tecnologías utilizadas para la implementación, como el lenguaje utilizado Java, o el lenguaje de consulta XPath utilizado para obtener información de las fuentes externas. Finalmente veremos también el entorno de desarrollo en el que se ha trabajado, Netbeans.

2.1. Arquitectura de Mediador

La consulta a los distintos portales web de venta productos tecnológicos, tiene que ser transparente para el usuario. El objetivo es que el usuario no perciba que está consultando en diferentes portales, tiene que tener la sensación de consultar solo uno. Para realizar este sistema lo más apropiado es utilizar una arquitectura de Mediador, cuyos componentes se ajustan perfectamente para realizar la función anteriormente descrita.

Esta arquitectura permite implementar de una manera sencilla y eficiente el proceso de extracción de información procedente de múltiples y variadas fuentes heterogéneas estructuradas o semi-estructuradas, así como el proceso de combinación de toda esta información y su transformación para obtener un producto informativo entregable al usuario final.

La arquitectura mediador es una arquitectura muy flexible ya que permite la integración de datos de cualquier fuente, ya sea servidores, bases de datos, documen-

tos... así como los ajustes necesarios para la implementación de transformadores de dicha información. El objetivo es conseguir un acceso unificado a todos los datos extraídos.

2.1.1. Funcionamiento básico de la arquitectura

A continuación pasamos a explicar en detalle las características de esta arquitectura y como se ha ido adaptando al proyecto que nos ocupa.

La arquitectura Mediador está compuesta por tres niveles: Nivel de Usuario o de Aplicación, Nivel de Integración y Nivel de Fuente de Datos, estos tres niveles pueden ser contemplados en la Figura 1, que muestra el flujo de peticiones realizadas desde el nivel más externo (Nivel de Usuario) hacia el nivel más interno (Nivel de Fuente de Datos).

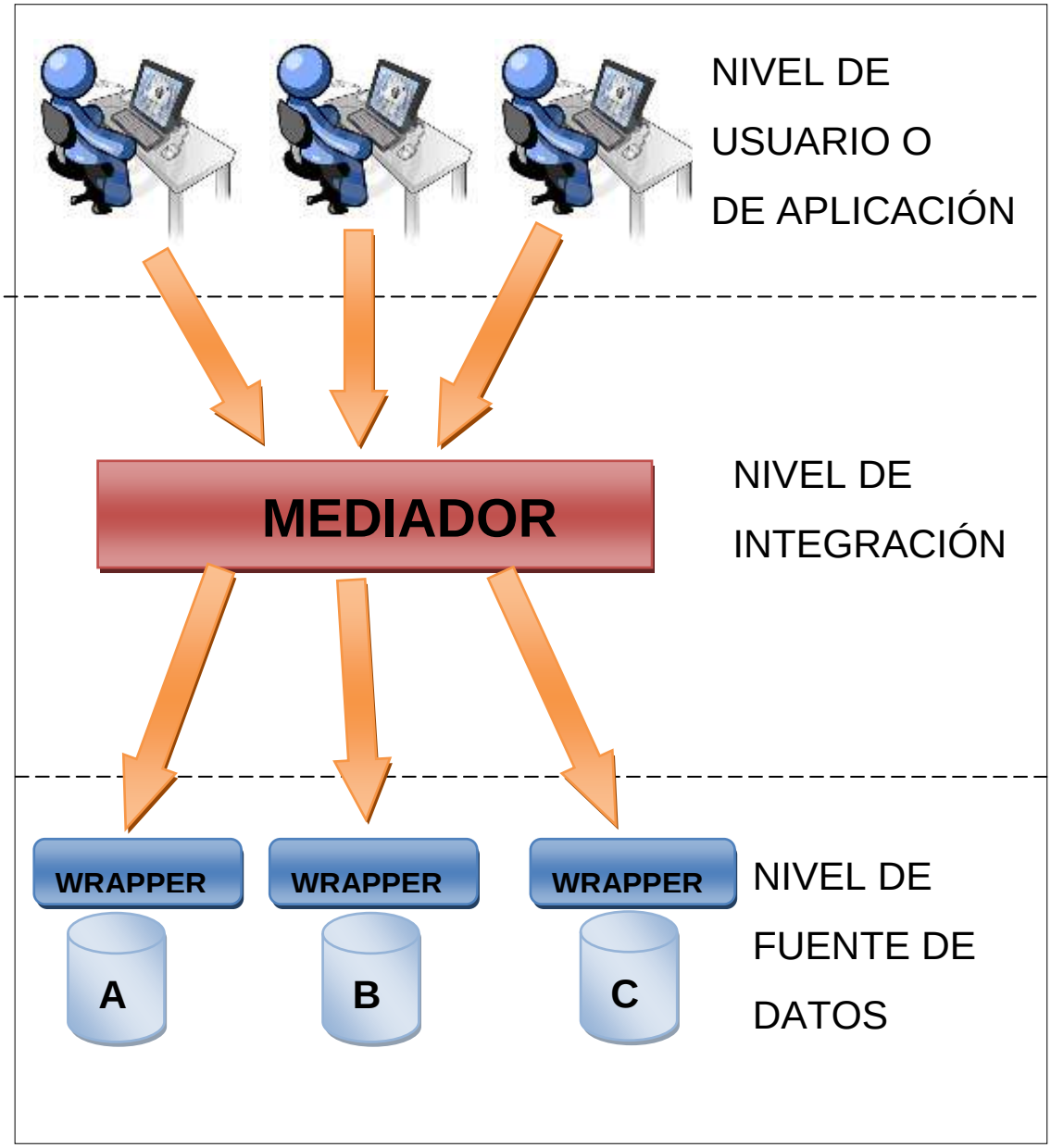

Figura 1: Arquitectura. Flujo de entrada de datos

Como se puede observar en la Figura 1, el Nivel de Usuario o de Aplicación está formado por la interfaz con la que los usuarios del sistema (las personas que lo van a utilizar) van a comunicarse con la aplicación. Estos usuarios al realizar una búsqueda, lo que están realizando es una petición al Nivel de Integración con la información de lo que ellos quieren encontrar. El Nivel de Usuario es el medio comunicador entre los usuarios y el Nivel de Integración.

En el Nivel de Integración la petición del usuario es recogida por la entidad Mediador, que analiza la petición y la transforma en diferentes peticiones individualizadas específicamente para cada uno de los diferentes wrappers o fuentes de informa-

ción externas a las que se consultará. Una vez transformadas se envían individualmente el formato apropiado al Nivel de Fuente de Datos.

En el Nivel de Fuente de Datos, se encuentran los distintos wrappers, que son las entidades que se encargarán de buscar información en las fuentes externas. Cada wrapper está asociado a una fuente de información y recibirá una petición específica del mediador, con la información adecuada para esa fuente concreta. Esta petición será procesada por el wrapper con el objetivo de extraer la información pedida de dicha fuente.

En la Figura 1 y en las explicaciones posteriores hemos visto cómo se realiza el flujo de peticiones en esta arquitectura desde el usuario hasta los distintos servidores. Veamos ahora como se realiza el flujo de información contrario. Es decir que es lo que ocurre una vez que los distintos wrappers del Nivel de Fuente de Datos han extraído y procesado la información de las fuentes hasta que llega en formato homogéneo al usuario. En la Figura 2 se observa este flujo de información

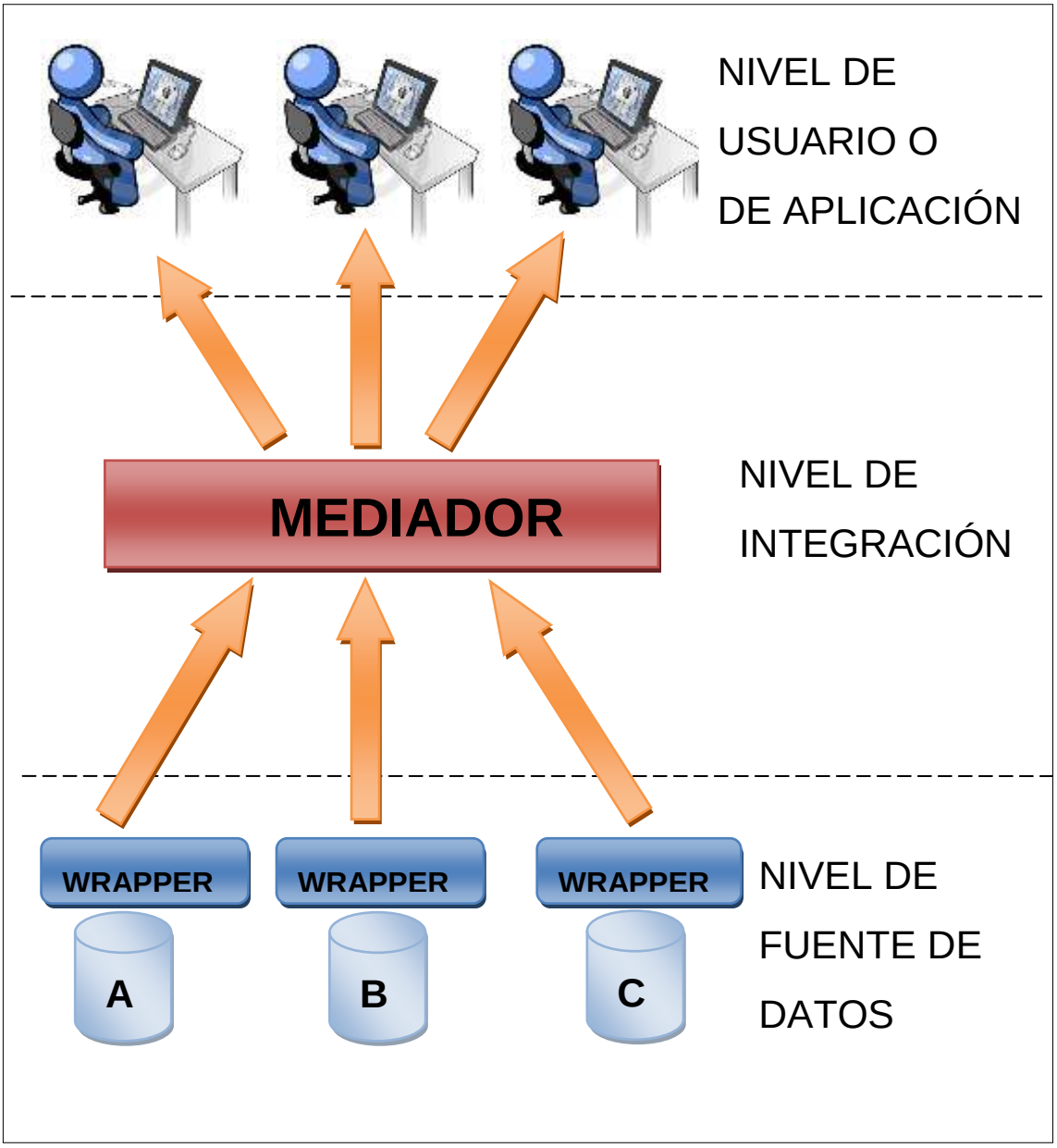

Figura 2: Arquitectura. Flujo de salida de datos

Una vez extraída la información de cada fuente asociada, los wrappers envían la información en el formato específico de cada una de ellas, hacia el Nivel de Integración.

En este nivel, se reciben todas las colecciones de productos de cada uno de los wrappers, se analiza esta información y se transforma a un formato unificado común, para después enviarla al Nivel de Usuario, donde el usuario de la aplicación observará esta información a través de la interfaz de la misma.

Este sería el funcionamiento básico de la arquitectura Mediador, el paso de peticiones e información desde que el usuario realiza una petición de búsqueda hasta que recibe la información requerida.

2.1.2. Ejemplo del funcionamiento de la arquitectura

Un usuario de la aplicación está interesado en buscar televisores de 42 pulgadas, con tecnología LCD. Para ello el usuario ejecuta la aplicación y se va a la pestaña de televisores. En esta pestaña se le mostrará las diferentes características de los televisores en listas desplegables para que el usuario pueda seleccionar las que le interesan. El usuario elige estas opciones, y pulsa en el botón buscar.

A partir de aquí la información introducida en el Nivel de Usuario es transmitida al Nivel de Mediador, en este nivel se creará una cadena de texto específica para cada tienda a buscar. Por ejemplo para la tienda Amazon.com, se creará la cadena "TV LCD 42-Inch" ya que de esta manera se suelen representar en Amazon el nombre de las televisiones almacenadas. Por el contrario para Optize.es la cadena de texto sería:" TV LCD 42". Estas cadenas son enviadas a cada uno de los wrappers del Nivel de Fuente de Datos.

En el nivel de Fuente de Datos cada wrapper navega hacia su portal web asociado y realiza una búsqueda en él, introduciendo la cadena de texto específica que le ha llegado del Nivel de Integración. La búsqueda se realiza y los wrappers extraen la información de los resultados. Aquí se completaría el flujo de ida de la información.

Durante el ciclo de vuelta, los resultados obtenidos en cada servidor son devueltos al Nivel de Integración donde se transforman a un formato homogéneo. Estos resultados finales son transmitidos al Nivel de Usuario y son mostrados en la interfaz gráfica donde el usuario puede ver el resultado de la búsqueda que él inició.

Este sería el funcionamiento básico de la arquitectura Mediador, el paso de peticiones e información desde que el usuario realiza una petición de búsqueda hasta que recibe la información requerida. Pasemos a ver en detalle cada uno de los niveles de la arquitectura mediador.

2.1.3. Niveles de la arquitectura

Veamos ahora una explicación más detallada de cada uno de los 3 niveles de los que se compone la arquitectura. Como hemos dicho anteriormente estos niveles son el Nivel de Usuario (o de Aplicación,) el Nivel de Integración y el Nivel de Fuente de Datos.

- **Nivel de Usuario o de Aplicación** este nivel es el más alto de los tres y está formado por la interfaz con la que interactuarán los usuarios finales con la aplicación. En él, se representa las interfaces de búsqueda de la aplicación y de muestra de resultados, donde la información de los productos será mostrada.

  Está compuesto por una interfaz en la que el usuario puede interactuar con la aplicación realizando peticiones de búsqueda y recibiendo información de la misma. Este nivel es el principio y el final de todo el proceso de búsqueda, desde el partirán las peticiones para realizar la búsqueda y a él llegarán la información final como resultado de la misma, en un formato unificado y adaptado para las necesidades del usuario.

- **Nivel de Integración:** este nivel es el que lleva casi todo la parte de procesamiento de información de la aplicación. Está compuesto por una entidad mediador que recibe las peticiones de los usuarios y realiza una función de adaptación de las mismas. Esta adaptación se realiza a un formato específico para cada entidad del Nivel de Fuente de Datos (wrappers), así se facilitará la búsqueda de información para cada servidor en particular.

  A este nivel también les llega la información extraída por el nivel inferior. Con lo que también se realiza en el Nivel de Integración la integración de esta información para ser enviada al nivel superior para que el usuario pueda recibirla de forma conjunta y homogénea, independientemente del número de portales donde se ha realizado la búsqueda.

  También cabe resaltar que como la información recibida del Nivel de Fuente de Datos puede ser muy extensa, en este nivel se procesa esta información a través de diferentes órdenes seleccionados por el usuario, para que se pueda quedar con la información más relevante.

- **Nivel de Fuente de Datos:** este nivel está formado por una entidad por cada portal del que queremos sacar información. Estas entidades como

hemos indicado antes se llaman wrappers (contenedores) y se encargarán de realizar las búsquedas en cada portal con la información específica que ha introducido el usuario y ha sido transformada por el Nivel de Integración. Los wrappers extraen la información de estos portales y la almacenan en un formato único, para enviarla posteriormente al nivel superior.

Los wrappers son partes muy costosas de implementar, pues cada uno es diferente debido a las notables diferencias de estructuración de código entre los portales de donde se quiere sacar información. Además hay que contar con el inconveniente de que la mayoría de las webs tienen la información semi-estructurada, lo que aumenta la complejidad para acceder a sus datos.

2.2. Selenium

2.2.1. Caracteristicas

Selenium es la tecnología obligatoria que hemos utilizado en este proyecto, ya que el proyecto consiste en la creación de un mashup utilizando esta tecnología. Se trata de un conjunto de herramientas de apoyo a la implementación de pruebas automáticas para aplicaciones basadas en web, que cuenta con las siguientes funcionalidades:

- Permite la grabación de los movimientos realizados por el usuario en el navegador. Estos movimientos puede ser reproducidos tantas veces como haga falta. Incluso pueden ser exportados a código en diferentes lenguajes de programación (característica de la que nos servimos para la utilización en el proyecto).
- Permite la localización de los distintos elementos HTML de cada web. Esto lo realiza utilizando librerías propias combinadas con el lenguaje XPath que se explicará en apartados posteriores. Estos elementos pueden ser por ejemplo una caja de búsqueda o cualquier enlace o imagen de la misma web.
- También permite comparar los resultados esperados de las pruebas con los resultados obtenidos realmente al realizarlas. (Esta característica no es útil para nuestro proyecto, pero si para el objetivo de Selenium).

2.2.2. Herramientas

Como hemos dicho antes, Selenium es un conjunto de herramientas, cada una de las cuales cumple un papel específico en la automatización de pruebas en aplicaciones web. Vamos a analizar cada una de estas herramientas, ofreciendo una explicación detallada de sus funcionalidades:

- **Selenium-IDE:** es la herramienta utilizada para construir conjuntos de pruebas. Su función es la de grabar las interacciones del usuario con el navegador. Permite grabar todas las acciones del usuario como los enlaces que pulse, los elementos que utilice, el texto que introduzca... como una serie de acciones a ejecutar sobre un conjunto de elementos HTML. Estas acciones quedan guardadas en un script y se pueden reproducir las veces deseadas. La presentación de esta aplicación es un plugin para el navegador Mozilla Firefox. Además de realizar las grabaciones de los casos también permite exportarlos a diferentes lenguajes (como C# o Java) y editar los mismos por si queremos ajustar algún detalle que no haya sido grabado.

- **Selenium-RC:** Selenium Remote Control ofrece una API con librerías para cada uno de los lenguajes permitidos por Selenium (HTML, Java, C#, Perl, PHP, Python, y Ruby). Permite al desarrollador de los casos de prueba usar de una forma flexible y extensible estos lenguajes de programación para desarrollar los tests. Además, debido a que los lenguajes soportados son de alto nivel, permite la integración de esta tecnología en cualquier proyecto de un entorno de programación de los lenguajes más utilizados actualmente.

  Se divide en dos partes:

  - Un servidor que automáticamente lanza el navegador y actuar como un proxy HTTP para realizar peticiones.
  - Un cliente que contiene las librerías de los lenguajes de programación comentados anteriormente.

- **Selenium-Grid:** esta herramienta no es usada para este proyecto, pero aun así la comentaremos para que el lector se haga una idea de la misma. Selenium-Grid permite que los conjuntos de pruebas realizados por Selenium-RC sean instanciados en distintos entornos. Así nos permite ejecutar varias instancias de Selenium-RC en distintos sistemas operati-

vos a la vez y con distintas configuraciones del navegador. Esto permite realizar tests en paralelo.

2.2.3. Selenium IDE: Interfaz

A continuación se mostrará cómo es la interfaz de la herramienta Selenium-IDE desde la que hemos grabados la interacción con el navegador necesaria para realizar las búsquedas en cada página web fuente contemplada.

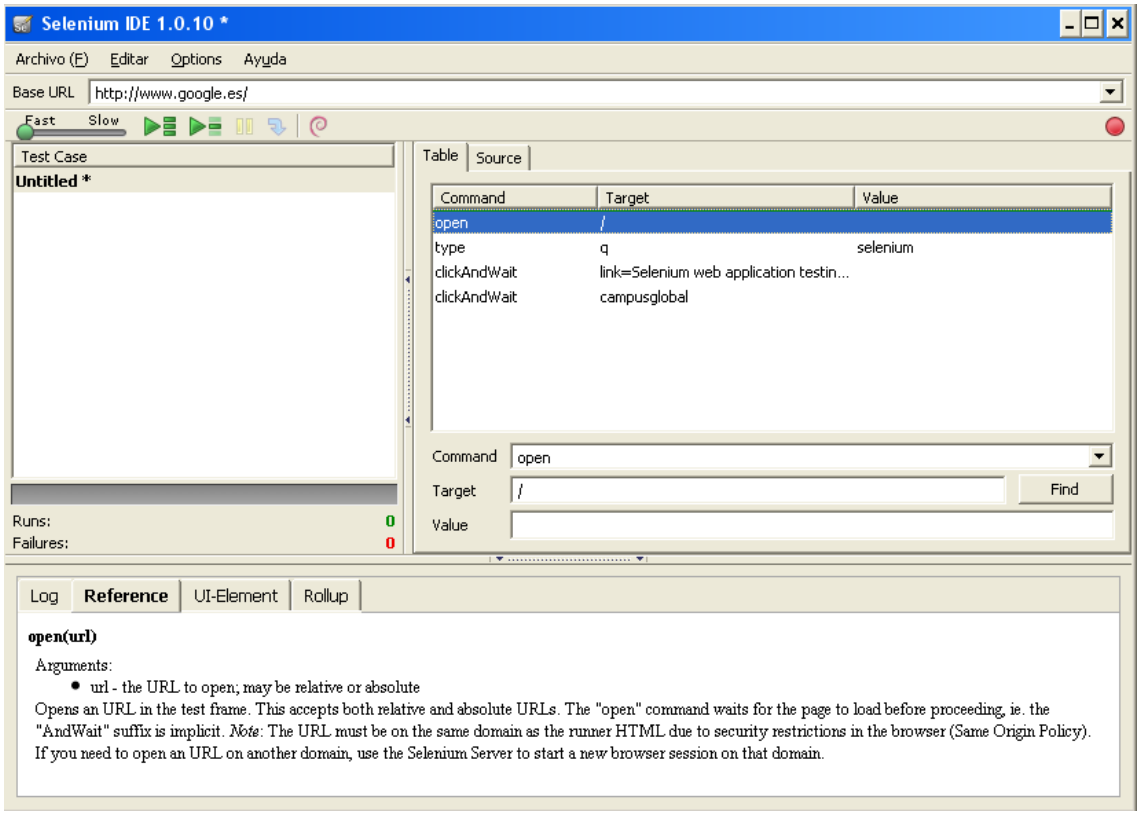

Figura 3: Interfaz de Selenium-IDE

Como podemos ver en la Figura 3, en Selenium-IDE hay un botón en la derecha de la barra de herramientas que sirve para grabar el comportamiento del navegador. Cuando está activado, la aplicación grabará automáticamente todas las interacciones que el usuario realice con el navegador.

Estas iteraciones se grabarán en forma de comandos, objetivos y valores (como se refleja en la pestaña “Table”, de la parte derecha de la interfaz). Así por ejemplo una acción puede estar compuesta por el comando “Type” (que significa que escribimos un texto en la web), el objetivo “Q” que en este caso representa el elemento de la caja de búsqueda de Google (véase que la URL actual es la de Google, en la

parte superior de la interfaz), y por último el valor sería el texto que escribimos en ella que en este caso es “Selenium”. Con lo cual este comando representaría la introducción del texto “Selenium” en la caja de búsqueda de Google.

En la pestaña de la izquierda se guardan las grabaciones, cada una de ellas forma un conjunto de pruebas por separado, aunque luego, editándolas, se pueden unir varios conjuntos. Desde esta pestaña se pueden ejecutar cada uno de estos conjuntos y ver los fallos producidos en esas pruebas.

Por ultimo en la parte inferior de la pantalla hay un conjunto de pestañas en las que se ofrecen explicaciones sobre comandos o elementos de la interfaz, así como un histórico de todo lo realizado por el usuario en la aplicación.

Aplicación práctica al proyecto

En cuanto a la aplicación al proyecto que nos ocupa, se ha utilizado la tecnología Selenium para adquirir la funcionalidad de lanzar el navegador Web desde nuestra aplicación, introducir la dirección del portal del que queremos extraer la información, seleccionar el elemento de caja de búsqueda, introducir el texto de búsqueda que el usuario ha introducido previamente en nuestra aplicación y realizar la búsqueda, extrayendo posteriormente los resultados obtenidos.

Así el procedimiento ha sido grabar un caso de prueba realizando una búsqueda en cada portal web con la herramienta Selenium-IDE, exportarlo a lenguaje Java, y mediante la combinación de los mismos con otras funciones de la API proporcionada por Selenium-RC, introducirlo en el proyecto Java creado, para realizar la función de navegación, búsqueda y extracción de información de los portales webs seleccionados.

2.2.4. Otras alternativas

A la hora de plantear el proyecto, se barajó una alternativa a Selenium como tecnología a emplear, la tecnología Ubiquity. En las siguientes líneas vamos a realizar una descripción de la misma.

Ubiquity es una tecnología desarrollada por Mozilla Labs durante los años 2008 y 2009 (actualmente se encuentra parado su desarrollo). Se trata de una extensión para Mozilla Firefox que está compuesta por una colección de comandos que actúan como un mashup de servicio webs. Permite a los usuarios obtener información de estos servicios, a través de los comandos y relacionarla. Estos comandos son los que se utilizarían para obtener la información en los mashup que se pueden implementar con esta tecnología. Entre otras Ubiquity permite insertar mapas en cualquier lugar, traducir páginas web…

La extensión para Mozilla Firefox, presenta una interfaz parecida a una línea de comandos. Esta interfaz está representada en la Figura 4. Como se puede ver, solo tiene una línea de comandos en la parte superior de la misma, donde el usuario puede escribir el comando deseado.

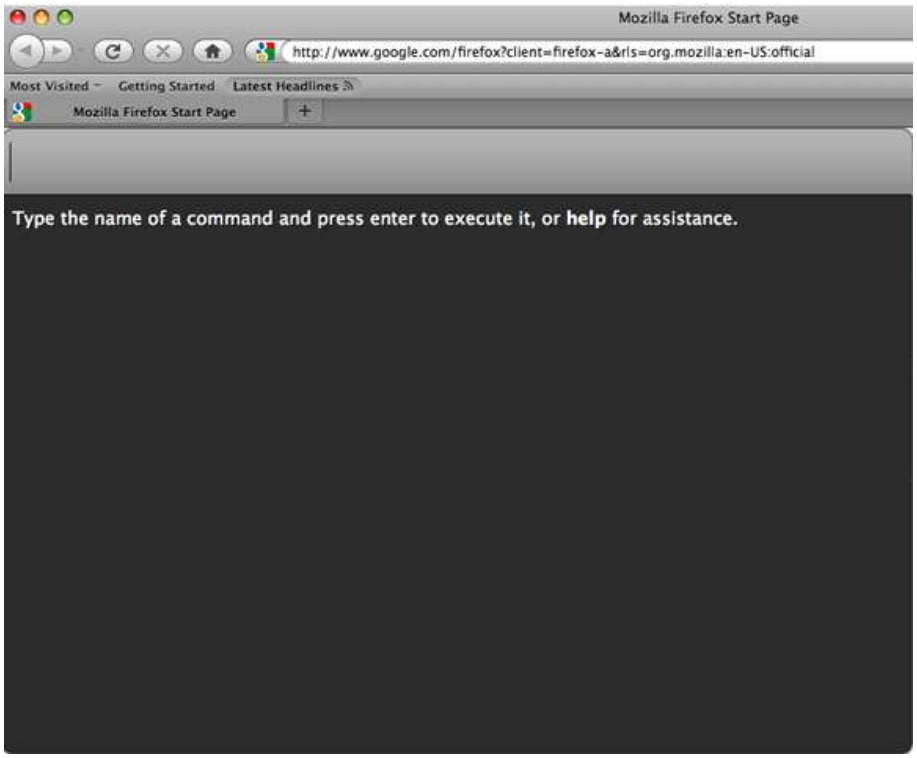

Figura 4: Interfaz de Ubiquity

Veamos un ejemplo de la creación de un comando, para entender mejor su funcionamiento:

Imaginemos que queremos crear un comando que escriba un email a través de una cuenta de correo electrónico de Google (Gmail). Solo tenemos que escribir en la línea de comando de la interfaz de Ubiquity la palabra “email” e inmediatamente nos saldrá la sugerencia del comando, que podemos ver en la Figura 5.

Figura 5: Creación de un comando en Ubiquity

Esto indica que existe un comando “email (message) (to contact)” que acepta dos parámetros: el mensaje y el contacto de Gmail a quien quiere enviarse el correo electrónico. Con lo cual si queremos crear por ejemplo un comando que envía el mensaje “hello” a un contacto llamado “Chris”. La sintaxis sería la siguiente: "email hello to chris". Esto sería entendido como ubiquity de navegar hasta nuestra cuenta de Gmail, hacer clic en “Compose new email” e introducir el destinatario y el texto del mensaje para finalmente enviarlo. Estos comandos son los que se utilizarían para obtener la información en los mashup que se pueden implementar con esta tecnología.

Esta alternativa fue descartada por dos motivos fundamentales: El primero es que actualmente está parado su desarrollo, así como su soporte y el segundo es la escasa documentación que existe al respecto, lo que dificultaría la tarea de desarrollo del proyecto.

2.3. Java

Tras barajar diferentes posibilidades a la hora de elegir un lenguaje para realizar el proyecto, Java fue finalmente el lenguaje elegido. Más adelante explicaremos las razones de esta elección, pero primero vamos a realizar una descripción del mismo.

Java es un lenguaje de programación orientado a objetos, que toma un modelo de objetos simple. Fue creado por Sun Microsystems y uno de sus puntos fuertes es que es un lenguaje independiente de la plataforma, ya que se ejecuta en una máquina virtual, que puede ser instalada en cualquier sistema operativo. Con lo cual las aplicaciones desarrolladas en Java pueden ser ejecutadas en cualquier plataforma.

La mayoría de su sintaxis es muy parecida a la existente en C y C++, con la diferencia de que tiene un modelo de objetos más simple. Una de las principales mejoras respecto a estos es que evita tratar de forma directa con punteros y con memoria, eliminando así una gran parte de los errores respecto a estos lenguajes que si tratan directamente con estas herramientas de bajo nivel.

En cuanto a las razones por las que se eligió este lenguaje para la implementación de la aplicación, son varias:

- La primera razón por la cual me decanté por Java, fue por la independencia de la plataforma. A la hora de programar es más sencillo poder realizarlo desde cualquier sistema operativo, solo hacía falta tener la

máquina virtual para ejecutar y probar la aplicación, lo que da mucha libertad para el desarrollador

Al igual, al poderse ejecutar las aplicación en cualquier sistema operativo, el número de potenciales usuarios aumenta con respecto a la utilización de un lenguaje cuyas aplicaciones solo se pudieran ejecutar en una plataforma.

- La segunda razón es que personalmente estoy muy familiarizado con el mismo y me resultaba más sencillo este lenguaje que cualquier otro, ya que conozco perfectamente su estructura y he desarrollado muchas otras aplicaciones utilizándolo.

- La tercera y última razón es que el entorno de programación utilizado para el desarrollo del proyecto (Netbeans) es muy conocido por mí. La mayoría de proyectos software que he realizado los he realizado utilizando este entorno de programación, así conozco muy bien todas sus funciones y sus posibilidades.

  Esto me facilitó mucho a la hora de instalar las nuevas librerías o hacer una migración de tipo de proyecto al implementar la interfaz Java. Este entorno de programación lo explicaremos más adelante.

A continuación describiremos por encima las demás opciones contempladas para desarrollar la aplicación, para que el lector se haga una idea de las alternativas barajadas:

- **C**#: este lenguaje fue creado por Microsoft como parte de su plataforma .NET. Es un lenguaje orientado a objetos cuya sintaxis deriva de C/C++. El sistema de objetos es muy similar a Java. El entorno de programación más utilizado para desarrollar aplicaciones en este lenguaje es Microsoft Visual Studio, con el cual también estoy muy familiarizado. Por su sencillez y el conocimiento previo adquirido, fue contemplado como la única alternativa seria a Java para realizar el proyecto.

- **Python:** este lenguaje es administrado por la Python Software Foundation y es de código abierto. En cuanto a sus características podemos decir que es un lenguaje interpretado y multiplataforma, además aúna varias características importantes: es orientado a objetos, de programación

imperativa y también de programación funcional. El motivo por el cual no se eligió este lenguaje en lugar de Java es por el desconocimiento del autor sobre el mismo. Aunque fue recomendado para utilizarse, el completo desconocimiento del lenguaje obligaba a aprender su sintaxis y estructura antes de empezar la implementación, lo que retrasaba más esta etapa.

Aplicación práctica al proyecto

El lenguaje Java ha sido la base para realizar el proyecto, pues la aplicación está desarrollada completamente en este lenguaje, con la salvedad de la inclusión del lenguaje XPath para acceder a elementos en los documentos HTML de los distintos portales.

Así desde la capa más externa (la interfaz gráfica) hasta la parte más interna ha sido implementada en el lenguaje Java, siempre con ayuda de las librerías que Selenium dispone para este lenguaje.

2.4. XPath

El lenguaje de consulta XPath permite construir expresiones que pueden recorrer y procesar un documento XML, teniendo en cuenta la estructura jerárquica de dicho documento. Nos permite acceder a los distintos elementos de la estructura XML de una página web. Cada expresión construida con este lenguaje recorre y encuentra el elemento deseado dentro de un documento XML. Su búsqueda se basa en las etiquetas identificativas de estos elementos.

Un ejemplo de expresión XPath es: “/bookstore/book[price>35.00]/title”. Esta expresión seleccionará todos los elementos “title” de los elementos “book” del elemento “bookstore” que tengan un precio mayor a 35.

Un inconveniente de este lenguaje es que accede a una ruta fija del elemento en la web, con el inconveniente de que si se cambia esta ruta, el elemento no puede ser encontrado, ya que no se detectan estos cambios. En este caso habría que cambiar la expresión XPath con la nueva ruta. Evidentemente, estos casos serían menos probable si las páginas webs tuvieran su información bien estructurada y dotarán a sus elementos de etiquetas identificativas para facilitar su búsqueda. Aun así se ha intentado utilizar expresiones lo más robustas posibles para minimizar el impacto en la aplicación de un cambio producido en la estructura de uno de los portales contemplados.

Aplicación práctica al proyecto

En el proyecto se ha utilizado la tecnología XPath para acceder a la información que queremos extraer de cada página web, así podemos acceder a los distintos elementos XML que se encuentran dentro del código HTML de cada portal seleccionado.

Una vez que hemos realizado la búsqueda, nos aparecerá la página web con los resultados de los productos que queremos extraer. Con lo cual tenemos que acceder a cada uno de los elementos de cada resultado (imágenes, cuadros de textos, enlaces...), utilizando una expresión XPath. Una vez que le hemos encontrado, extraemos su valor mediante el uso de las librerías de Selenium y lo almacenamos en variables para su posterior procesamiento.

2.5. Entorno de desarrollo

En cuanto al entorno de programación a utilizar, la verdad que fue lo que más dudas me planteó, ya que me sentía muy familiarizado con dos de los entornos más utilizados para el lenguaje de programación Java. Estos son Netbeans y Eclipse. Finalmente una de las razones de la elección de Netbeans fue la posibilidad de implementar interfaces gráficas desde el mismo entorno, sin tener que instalar plugins adicionales. Pasemos a detallar las características del entorno de programación elegido.

2.5.1. Netbeans

Netbeans es un entorno de desarrollo de código abierto realizado especialmente para trabajar con lenguaje Java. El fundador del mismo es Sun MicroSystem quien lo creo en el año 2000. Al ser de código libre el coste del uso del mismo es gratuito. Pasemos a ver las ventajas y los inconvenientes que presenta utilizar Netbeans como entorno de desarrollo.

Ventajas de utilizar Netbeans:

- La interfaz de Netbeans es muy intuitiva y realmente rápida a la hora de crear aplicaciones Java. Ya que esta específicamente diseñado para este lenguaje. Así sus diferentes vistas (código, archivos de proyecto, propiedades de elementos, métodos de clases…) permiten al desarrollador

en un simple vistazo tener al alcance todas los elementos que componen el proyecto.

- Trae una herramienta gráfica incorporada para realizar interfaces Java utilizando las clases para AWT y SWING para ello. Así crear una interfaz simple en Java con Netbeans se convierte en una tarea sencilla. También se pueden configurar las propiedades de los elementos gráficos desde la propia interfaz gráfica, sin necesidad de meterse en el código de la aplicación.

- Tiene un repositorio de plugins desde el cual es fácil encontrar el que se desea fácilmente sin tener que ir a buscarle fuera del programa. Desde la propia interfaz del mismo se puede obtener acceso a este repositorio.

- Hay mucha documentación sobre este entorno. Al ser uno de los entornos más utilizados, en la red hay numeroso material acerca de las posibilidades que Netbeans ofrece, así como tutoriales paso a paso para realizar cada una de las funcionalidades de este entorno.

Desventajas de utilizar Netbeans:

- El número de plugins del repositorio de Netbeans es actualmente inferior al ofrecido por su gran competidor: Eclipse.

- El rendimiento de este entorno es algo más bajo que Eclipse. Sobre todo para procesadores un poco antiguos, se ralentiza la carga del código de las clases, así como la importación automática de clases. Aunque esto se nota con mayor diferencia cuando se está implementando la interfaz gráfica.

2.5.2. Otras alternativas: Eclipse

Como hemos dicho al principio de este punto, la única alternativa a Netbeans que se valoró fue la del entorno de programación Eclipse. Aunque las similitudes entre ambos son muchas, existen algunas diferencias entre ellos. Pasemos a valorar algunas de las características de este entorno de programación.

- Soporta muchos lenguajes de programación como Java, C++ o PHP, así se puede utilizar el mismo entorno para diferentes proyectos software con distintos lenguajes. Esto es una ventaja general de Eclipse que no es interesante para nuestro proyecto, al estar realizado en Java.

- Tiene muchos plugins que pueden ser fácilmente instalados. Esto es muy útil a la hora de realizar pruebas unitarias con algún sistema específico. O por ejemplo a la hora de realizar la interfaz gráfica, ya que Eclipse posee bastantes plugins para esta función.

- Hay otros entornos de programación cuya interfaz y diseño se basa en Eclipse (Como por ejemplo Adobe Flash Builder). Así si estás acostumbrado a utilizar alguno de estos entornos te será mucho más fácil utilizar Eclipse al sentirte familiarizado con él.

- Generalmente para las distintas versiones de Eclipse, el inicio es muy lento en comparación con Netbeans. Aunque este es un inconveniente menor porque la carga de la aplicación se realiza una vez.

- Tiene bastantes menos funcionalidades de serie que Netbeans, pero al existir un número elevado de plugins para Eclipse, son cubiertas por estos. Así en la mayoría de los aspectos, si contamos con los plugins, Eclipse ofrece un mayor número de funcionalidades.

La decisión final de Netbeans se basó en dos razones muy poco significativas, ya que la elección de Eclipse tampoco me hubiera supuesto mucho esfuerzo. La primera razón es que es más sencillo crear una interfaz de usuario para Java con este entorno, ya que trae la herramienta incorporada y hay más documentación sobre ella. La segunda razón es que en el momento de empezar este proyecto tenía mejor conocimiento sobre Netbeans ya que lo acababa de utilizar para otros proyectos recientemente realizados.

En definitiva, por la sencillez y comodidad que me suponía empezar a trabajar con él, me decanté por Netbeans como entorno de desarrollo para la implementación de este proyecto.

3. Análisis

Los siguientes apartados están relacionados con el análisis del proyecto. En cuanto a esta etapa del desarrollo, veremos una especificación del problema a resolver y los requisitos establecidos para este proyecto. Para completar el análisis del mismo, se mostrarán todos los posibles casos de uso a través de sus diagramas correspondientes

3.1. Especificación del problema

La función principal de la aplicación realizada en el proyecto es la de permitir realizar una búsqueda en diferentes portales web de venta de artículos tecnológicos. Con esto se permite agrupar en un mismo lugar la información recogida de varios portales de venta online de este tipo de productos y así se le puede ofrecer al usuario en un único sitio toda esta información recopilada en formato unificado y homogéneo.

Cumpliendo este objetivo el usuario obtendrá toda esta información de una manera más rápida que si tuviera que realizar una búsqueda individual en cada página, además se podrá beneficiar de una mejor presentación de los resultados conjuntos, ya que el formato será homogéneo e independiente de la fuente de donde provenga la información.

Esta búsqueda ofrecerá un conjunto de resultados, el cual se podrá ordenar por distintos factores o atributos. Estos resultados deben permitir al usuario poder navegar directamente hacia la página de compra del producto para realizar el pago directamente desde la página Web de la empresa que lo oferta.

Además, en la información mostrada de cada producto se debe mostrar los atributos más relevantes para el usuario que desee comprarlo, estos son: Nombre, precio, foto y tienda del producto (todo ello junto con el enlace a la página que permitirá al usuario adquirir el producto).

Un objetivo secundario, pero no por ello menos deseado, de la aplicación, es que sea lo más específica posible. Por esta razón la búsqueda debe poderse hacer de forma personalizada para los diferentes tipos de productos. Así, existirá un método de búsqueda genérica en el que el usuario puede buscar cualquier producto introduciendo la cadena de texto que el desee, pero también existirá un método de búsqueda específico para cada tipo de producto tecnológico que hemos contemplado, en el que el usuario puede seleccionar las características del producto. Para este tipo de búsqueda, se han contemplado las familias de productos tecnológicos más relevantes, como los ordenadores personales, las impresoras o las cámaras fotográficas.

3.2. Requisitos

En esta sección especificaremos mediante una tabla los requisitos especificados que deben de ser cumplidos por nuestra aplicación. Estos requisitos aunque se pueden deducir de la especificación del problema, van a ser expuestos de manera formal.

Con el objetivo de realizar una presentación clara de los mismos, los presentamos en un formato sencillo y bien definido. El formato elegido para representarlos es el de la plantilla de la Tabla 1.

Identificador		**Tipo**	
Nombre	Un nombre corto y descriptivo para el requisito.		
Descripción	Una descripción detallada para el requisito.		
Verificabilidad	Si/No	**Criticidad**	1 a 5
Esencialidad	Si/No	**Deseabilidad**	1 a 5

Tabla 1: Plantilla de requisitos

Los campos de la tabla mostrada en la tabla anterior deben ser rellenados con la siguiente información:

- **Identificador**: un único y descriptivo código que identificará al requisito de manera univoca respecto a otros.
- **Tipo**: El tipo del requisito, puede ser: “Funcional”, “Arquitectura”, “Escalabilidad”, “Rendimiento”, “Seguridad”, “Tecnológico”, “Usabilidad” o de “Interfaz”.
- **Nombre**: un nombre corto y descriptivo del requisito para que el lector pueda hacerse una idea clara acerca del propósito del mismo.
- **Descripción**: una descripción detallada del requisito. En el que se explique el objetivo del mismo.
- **Verificabilidad:** si el requisito puede ser probado en el sistema. Este valor puede ser “Si” o “No”.
- **Criticidad**: el nivel de criticidad que tiene el requisito en el momento de especificarlo. El valor de este campo debe ser: “1”, “2”, “3”, “4” o “5”, donde “1” significa una baja criticidad y “5” una alta criticidad.

- **Esencialidad:** si el requisito es esencial o no para el correcto funcionamiento del sistema. Este valor puede ser “Si” o “No”.
- **Deseabilidad**: si el requisito no es esencial, este campo determina el nivel de deseabilidad que tiene El valor de este campo debe ser: “1”, “2”, “3”, “4” o “5”, donde “1” significa una baja deseabilidad y “5” una alta deseabilidad.

Sin más dilación nos disponemos a especificar los requisitos de usuario del proyecto:

Identificador	RU-001	**Tipo**	Funcional
Nombre	Búsqueda		
Descripción	El sistema debe ser capaz de realizar una búsqueda en diferentes portales webs que serán elegidos por el usuario.		
Verificabilidad	Si	**Criticidad**	5
Esencialidad	Si	**Deseabilidad**	

Tabla 2: RU-001: Búsqueda

Identificador	RU-002	**Tipo**	Funcional
Nombre	Búsqueda genérica		
Descripción	El sistema debe ser capaz de realizar una búsqueda con el texto a introducir por el usuario.		
Verificabilidad	Si	**Criticidad**	5
Esencialidad	Si	**Deseabilidad**	

Tabla 3: RU-002: Búsqueda genérica

Identificador	RU-003	**Tipo**	Funcional
Nombre	Búsqueda específica		
Descripción	El sistema debe ser capaz de realizar una búsqueda mediante la selección de características de productos específicos en la interfaz.		
Verificabilidad	Si	**Criticidad**	5
Esencialidad	No	**Deseabilidad**	5

Tabla 4: RU-003 Búsqueda específica

Identificador	RU-004	**Tipo**	Funcional
Nombre	Selección de fuentes		
Descripción	El sistema debe ser capaz de permitir al usuario seleccionar los portales donde desea realizar su búsqueda.		
Verificabilidad	Si	**Criticidad**	5
Esencialidad	Si	**Deseabilidad**	

Tabla 5: RU-004 Selección de fuentes

Identificador	RU-005	**Tipo**	Funcional
Nombre	Visualización de resultados		
Descripción	El sistema debe ser capaz de mostrar los resultados obtenidos de la búsqueda en la interfaz gráfica.		
Verificabilidad	Si	**Criticidad**	5
Esencialidad	Si	**Deseabilidad**	

Tabla 6: RU-005 Visualización de resultados

Identificador	RU-006	**Tipo**	Funcional
Nombre	Navegación entre resultados		
Descripción	El sistema debe ser capaz de permitir al usuario navegar entre los distintos resultados obtenidos en la búsqueda.		
Verificabilidad	Si	**Criticidad**	5
Esencialidad	Si	**Deseabilidad**	

Tabla 7: RU-006 Navegación entre resultados

Identificador	RU-007	**Tipo**	Funcional
Nombre	Ordenación de resultados		
Descripción	El sistema debe ser capaz de permitir al usuario ordenar los resultados por los criterios de precio, relevancia y portales.		
Verificabilidad	Si	**Criticidad**	5
Esencialidad	No	**Deseabilidad**	5

Tabla 8: RU-007 Ordenación de resultados

Identificador	RU-008	**Tipo**	Funcional
Nombre	Búsquedas sucesivas.		
Descripción	El sistema debe ser capaz de realizar una nueva búsqueda tras haber finalizado la búsqueda actual.		
Verificabilidad	Si	**Criticidad**	5
Esencialidad	No	**Deseabilidad**	5

Tabla 9: RU-008 Búsquedas sucesivas

Identificador	RU-009	**Tipo**	Funcional
Nombre	Acceso a páginas de compra		
Descripción	El sistema debe ser capaz de permitir al usuario acceder a las páginas de compra de los productos mostrados en los resultados.		
Verificabilidad	Si	**Criticidad**	5
Esencialidad	Si	**Deseabilidad**	

Tabla 10: RU-009 Acceso a páginas de compra

Identificador	RU-010	**Tipo**	Funcional
Nombre	Visualización de las principales características de los productos		
Descripción	El sistema debe ser capaz de permitir al usuario visualizar las principales características de un producto como son: Nombre, descripción, precio, fotografía y portal donde se oferta.		
Verificabilidad	Si	**Criticidad**	5
Esencialidad	Si	**Deseabilidad**	

Tabla 11: RU-010 Visualización de las principales características

Identificador	RU-011	**Tipo**	Funcional
Nombre	Productos específicos.		
Descripción	El sistema debe ser capaz de permitir realizar al usuario la búsqueda específica de las principales familias de productos tecnológicos del mercado.		
Verificabilidad	Si	**Criticidad**	4
Esencialidad	No	**Deseabilidad**	5

Tabla 12: RU-011 Productos específicos

Identificador	RU-012	**Tipo**	Funcional
Nombre	Características de productos específicos		
Descripción	El sistema debe ser capaz de permitir realizar al usuario una búsqueda específica de las principales familias de productos tecnológicos del mercado, contemplando como criterios las principales características de los mismos.		
Verificabilidad	Si	**Criticidad**	4
Esencialidad	No	**Deseabilidad**	5

Tabla 13: RU-012 Características de productos específicos

Identificador	RU-013	**Tipo**	Funcional
Nombre	Precios de los productos específicos		
Descripción	El sistema debe ser capaz de permitir realizar al usuario la búsqueda específica de las principales familias de productos tecnológicos del mercado, indicando un rango de precios de los mismos.		
Verificabilidad	Si	**Criticidad**	4
Esencialidad	No	**Deseabilidad**	5

Tabla 14: RU-013 Precios de los productos específicos

Identificador	RU-014	**Tipo**	Rendimiento
Nombre	Eficiencia y rapidez.		
Descripción	El sistema debe de ofrecer un rendimiento rápido y eficiente en sus búsquedas, con el objetivo de mejorar la ventaja en este aspecto a la búsqueda manual por parte del usuario.		
Verificabilidad	Si	**Criticidad**	5
Esencialidad	Si	**Deseabilidad**	

Tabla 15: RU-014 Eficiencia y rapidez

Identificador	RU-015	**Tipo**	Rendimiento
Nombre	Consumo de recursos		
Descripción	El sistema debe de ofrecer un rendimiento eficiente consumiendo un número bajo de recursos de la máquina en la que se esté ejecutando.		
Verificabilidad	Si	**Criticidad**	5
Esencialidad	No	**Deseabilidad**	5

Tabla 16: RU-015 Consumo de recursos

Identificador	RU-016	**Tipo**	Temporal
Nombre	Límite temporal en búsquedas		
Descripción	El sistema debe de realizar las búsquedas en un tiempo límite establecido por el usuario. Una vez sobrepasado este tiempo se mostrará un error considerándose un problema de conexión.		
Verificabilidad	Si	**Criticidad**	5
Esencialidad	No	**Deseabilidad**	5

Tabla 17: RU-016 Limite temporal en búsquedas

Identificador	RU-017	**Tipo**	Arquitectura
Nombre	Flexibilidad de la arquitectura.		
Descripción	La arquitectura que implementa el sistema debe de ser flexible con el objetivo de poder añadir más fuentes de búsqueda a la misma con el mínimo coste.		
Verificabilidad	Si	**Criticidad**	5
Esencialidad	No	**Deseabilidad**	5

Tabla 18: RU-017 Flexibilidad en la arquitectura

Identificador	RU-018	**Tipo**	Seguridad
Nombre	Seguridad hardware.		
Descripción	El sistema debe de ofrecer seguridad completa en todo momento a la máquina donde se está ejecutando. No suponiendo ningún riesgo para la misma.		
Verificabilidad	Si	**Criticidad**	5
Esencialidad	Si	**Deseabilidad**	

Tabla 19: RU-018 Seguridad Hardware

Identificador	RU-019	**Tipo**	Seguridad
Nombre	Seguridad de datos		
Descripción	El sistema debe de ofrecer seguridad completa en todo momento al usuario. No relevando ni almacenando en sitios externos los datos de las búsquedas que el usuario realiza.		
Verificabilidad	Si	**Criticidad**	5
Esencialidad	Si	**Deseabilidad**	

Tabla 20: RU-019 Seguridad de datos

Identificador	RU-020	**Tipo**	Tecnológico
Nombre	Independencia de la plataforma		
Descripción	El sistema debe de ser capaz de ejecutarse en cualquier sistema operativo, eliminando así los problemas de portabilidad de una plataforma a otra.		
Verificabilidad	Si	**Criticidad**	5
Esencialidad	No	**Deseabilidad**	5

Tabla 21: RU-020 Independencia de la plataforma

Identificador	RU-021	**Tipo**	Usabilidad
Nombre	Usabilidad en búsquedas		
Descripción	El sistema debe de ofrecer una usabilidad muy alta en la tarea de realizar las búsquedas. Así tiene que ser una tarea sencilla para el usuario realizar las distintas búsquedas desde la interfaz del sistema.		
Verificabilidad	Si	**Criticidad**	5
Esencialidad	No	**Deseabilidad**	5

Tabla 22: RU-021 Usabilidad en búsquedas

Identificador	RU-022	**Tipo**	Usabilidad
Nombre	Usabilidad en la muestra de resultados		
Descripción	El sistema debe de ofrecer una usabilidad muy alta en la tarea de visualización de resultados. Así tiene que ser una tarea sencilla para el usuario la tarea de visualización de resultados.		
Verificabilidad	Si	**Criticidad**	5
Esencialidad	No	**Deseabilidad**	5

Tabla 23: RU-022 Usabilidad en la muestra de resultados

Identificador	RU-023	**Tipo**	Usabilidad
Nombre	Usabilidad en la ordenación de resultados		
Descripción	El sistema debe de ofrecer una usabilidad muy alta en la tarea de ordenación de resultados. Así tiene que ser una tarea sencilla para el usuario la tarea de ordenación de resultados.		
Verificabilidad	Si	**Criticidad**	5
Esencialidad	No	**Deseabilidad**	5

Tabla 24: RU-023 Usabilidad en la ordenación de resultados

Identificador	RU-024	**Tipo**	Interfaz
Nombre	Interfaz simple		
Descripción	El sistema debe proveer a los usuarios una interfaz simple y bien estructurada para facilitarles la interacción con la aplicación.		
Verificabilidad	Si	**Criticidad**	5
Esencialidad	No	**Deseabilidad**	5

Tabla 25: RU-024 Interfaz simple

Identificador	RU-025	**Tipo**	Interfaz
Nombre	Interfaz búsqueda genérica		
Descripción	El sistema debe proveer a los usuarios una interfaz para la funcionalidad de la búsqueda genérica de productos.		
Verificabilidad	Si	**Criticidad**	5
Esencialidad	Si	**Deseabilidad**	

Tabla 26: RU-025 Interfaz búsqueda genérica

Identificador	RU-026	**Tipo**	Interfaz
Nombre	Interfaz búsqueda específica		
Descripción	El sistema debe proveer a los usuarios una interfaz para la funcionalidad de la búsqueda especifica de productos.		
Verificabilidad	Si	**Criticidad**	5
Esencialidad	No	**Deseabilidad**	5

Tabla 27: RU-026 Interfaz búsqueda específica

Identificador	RU-027	**Tipo**	Interfaz
Nombre	Interfaz muestra de resultados		
Descripción	El sistema debe proveer a los usuarios una interfaz para la funcionalidad de la muestra de resultados.		
Verificabilidad	Si	**Criticidad**	5
Esencialidad	Si	**Deseabilidad**	

Tabla 28: RU-027 Interfaz muestra de resultados

Identificador	RU-028	**Tipo**	Interfaz
Nombre	Interfaz ordenación de resultados		
Descripción	El sistema debe proveer a los usuarios una interfaz para la funcionalidad de la ordenación de resultados.		
Verificabilidad	Si	**Criticidad**	5
Esencialidad	No	**Deseabilidad**	5

Tabla 29: RU-028 Interfaz ordenación de resultados

Identificador	RU-029	**Tipo**	Interfaz
Nombre	Navegación por pestañas		
Descripción	El sistema debe proveer a los usuarios una interfaz organizada en pestañas, cada una de estas tendrá asociados unas funcionalidades.		
Verificabilidad	Si	**Criticidad**	5
Esencialidad	No	**Deseabilidad**	5

Tabla 30: RU-029 Navegación por pestañas

Identificador	RU-030	**Tipo**	Escalabilidad
Nombre	Sistema escalable		
Descripción	El sistema debe ser escalable, para permitir introducir nuevas funcionalidades en el mismo, como la introducción de nuevos wrappers asociados a nuevos portales web donde poder realizar una búsqueda en ellos.		
Verificabilidad	Si	**Criticidad**	5
Esencialidad	No	**Deseabilidad**	5

Tabla 31: RU-030 Sistema escalable

3.3. Casos de uso

En este apartado mostraremos los casos de uso de la aplicación, que son la secuencia de interacciones que se desarrollan entre el sistema y los usuarios del mismo (actores). Es decir, los distintos eventos que inicia el usuario y que son respondidos por el sistema con diferentes acciones.

Para la aplicación, como se puede ver en la siguiente figura, el número de casos de usos es reducido y estos son bastante sencillos, pues la aplicación solamente realiza búsquedas y ofrece resultados, estos resultados a su vez pueden ser ordenados. Con lo cual los casos de uso se limitan a los relacionados con las búsquedas (tanto especifica como genérica) como con los resultados (visualización y ordenación).

También se puede seleccionar en que portales se quiere realizar la búsqueda, tanto genérica como específica. Esto se realiza con el objetivo de simplificar la obtención de resultados de la búsqueda así como de la optimización en el rendimiento de la misma. Estos casos de uso quedan detallados en el diagrama siguiente de la Figura 6.

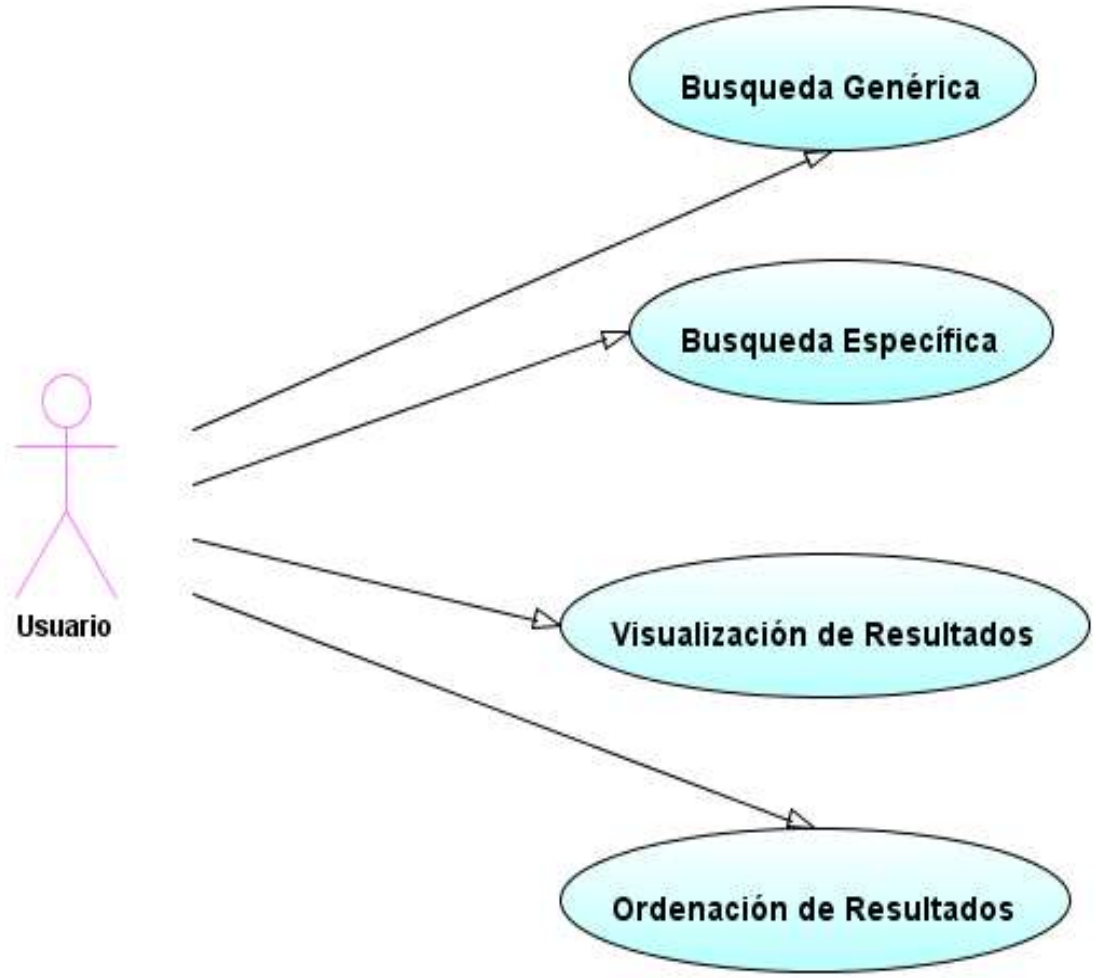

Figura 6: Casos de uso de la aplicación

A continuación vamos a especificar los casos de uso. Para ello lo haremos mediante la utilización de tablas cuyo formato permite explicar de forma clara y concisa cada uno de ellos. En la siguiente tabla se expone una plantilla del formato con el que se mostrarán.

Identificador	
Nombre	
Descripción	
Pre-Condiciones	
Flujo normal	

Tabla 32: Plantilla de casos de uso

Así, los campos de la tabla anterior tienen que ser rellenados con la siguiente información:

- **Identificador**: un único y descriptivo código que identificará al caso de uso de manera unívoca respecto a otros.
- **Nombre**: un nombre corto y descriptivo del caso de uso para que el lector pueda hacerse una idea clara acerca del propósito del mismo.
- **Descripción**: una descripción detallada del caso de uso. En el que se explique el objetivo del mismo.
- **Precondiciones**: precondiciones que se tienen que dar en el sistema para que se pueda llevar a cabo el caso de uso.
- **Flujo normal**: flujo de ejecución del caso de uso. Pasos que se necesitan dar para llevar a cabo el caso de uso.

3.3.1. Búsqueda genérica

Identificador	CU-001
Nombre	Búsqueda genérica
Descripción	El sistema debe permitir realizar una búsqueda genérica del producto que el usuario desee. Esto se realizará por medio de la introducción del texto descriptivo del producto por parte del usuario.
Pre-Condiciones	El sistema se debe de estar ejecutando normal y correctamente en la pantalla principal, correspondiente a la búsqueda genérica.
Flujo normal	El usuario introduce una cadena de texto en la pantalla ini-

	cial del sistema, selecciona los portales donde desee buscar y presiona el botón de “buscar”. El sistema empezará a realizar la búsqueda en los distintos portales y le devolverá al usuario los resultados de la misma.

Tabla 33: CU-001 Búsqueda genérica

3.3.2. Búsqueda específica

Identificador	CU-002
Nombre	Búsqueda específica
Descripción	El sistema debe permitir realizar una búsqueda específica del producto que el usuario desee, dentro de los productos contemplados para la misma. Esto se realizará por medio de la elección en un menú desplegable de las distintas características del producto.
Pre-Condiciones	El sistema se debe de estar ejecutando normal y correctamente, y se debe pulsar sobre una pestaña que llevan a la búsqueda específica de un determinado tipo de producto.
Flujo normal	El usuario selecciona el tipo de producto y sus características en la pantalla específica de ese tipo de producto en la aplicación, selecciona los portales donde desee buscar y presiona el botón de “buscar”. El sistema empezará a realizar la búsqueda en los distintos portales y le devolverá al usuario los resultados de la misma

Tabla 34: CU-002 Búsqueda específica

3.3.3. Visualización de resultados

Identificador	CU-003
Nombre	Visualización de resultados.
Descripción	El sistema debe permitir tras realizar una búsqueda genérica o específica, mostrar los resultados de la misma. Esto se hará en una interfaz de muestra de resultados en el que la información será almacenada de forma homogénea.
Pre-Condiciones	El sistema se debe de estar ejecutando normal y correcta-

	mente, y se debe de haber realizado una búsqueda anteriormente.
Flujo normal	El usuario realiza una búsqueda de un producto y cuando esta finaliza automáticamente se muestra la pantalla en la que se muestran los resultados obtenidos.

Tabla 35: CU-003 Visualización de resultados

3.3.4. Ordenación de resultados

Identificador	CU-004
Nombre	Ordenación de resultados
Descripción	El sistema debe permitir tras mostrar los resultados de una búsqueda, la posibilidad de ordenarlos basándose en los criterios de precio, relevancia y agrupación por portales.
Pre-Condiciones	El sistema se debe de estar ejecutando normal y correctamente, y se debe de haber realizado una búsqueda anteriormente, y estar en la pantalla de muestra de resultados.
Flujo normal	El usuario realiza una búsqueda de un producto y cuando esta finaliza automáticamente se muestra la pantalla en la que se muestran los resultados obtenidos. En ella, el usuario pulsara sobre la lista de criterios para ordenar los resultados, mostrándose automáticamente ordenados tras pulsar cualquiera de ellos.

Tabla 36: CU-004 Ordenación de resultados

4. Diseño

En los siguientes apartados comentaremos los aspectos referentes a la etapa de diseño del desarrollo del proyecto. En ellos explicaremos en profundidad la arquitectura utilizada, así como cada capa de la misma, para que el lector pueda tener un conocimiento bastante profundo sobre cómo se ha estructurado la aplicación. En el apartado final se explicará con detalle cómo se ha realizado el diseño de la interfaz.

4.1. Arquitectura

4.1.1. Arquitectura de Mediador

La arquitectura elegida para el desarrollo del proyecto ha sido la “Arquitectura de Mediador”. Esta arquitectura ya se ha explicado brevemente en el apartado 2.1, pero profundizaremos más sobre ella en este apartado.

La razón por la que se ha elegido esta arquitectura es porque permite ajustar de una manera sencilla y eficiente el proceso de extracción de información procedente de fuentes variadas, cuya información puede estar estructurada o no. Lo que quieres decir que podemos encontrar parte de la información sin una estructura claramente definida. Para ello tenemos unas entidades (wrappers) en la arquitectura cuya función es extraer esta información para cada portal en particular.

También esta arquitectura nos permite combinar toda esta información obtenida en el proceso de extracción, de manera que la podamos transformar para obtener un producto informativo unificado (con la misma estructura independientemente de la fuente de donde proceda).

A continuación mostramos la estructura de este producto informativo unificado que hemos contemplado, para que el lector se haga una idea de cómo está formado se incluirá un ejemplo de cómo se rellenaría cada campo de la estructura.

El formato de la estructura en la que guardamos los productos obtenidos de las búsquedas es la siguiente:

- **Nombre del producto:** se rellenará con el nombre con el que el portal que lo oferta, identifica al producto.
- **Precio:** se rellenará con el precio de venta del producto del portal que lo oferta.
- **Enlace del producto:** aquí se guardara el link del sitio web específico del producto en el portal donde se oferta. Desde este sitio web se podrá proceder directamente a su compra.

- **Enlace de la foto del producto:** es el link donde se almacena la foto del producto. Es utilizado posteriormente para mostrar esta foto en la pantalla del interfaz.
- **Portal:** nombre del portal en el que se oferta el producto.

Este producto informativo será el entregable al usuario final, el cual obtendrá la información de forma uniforme, y no tendrá que preocuparse por las diferentes características los productos de cada portal donde ha deseado buscar (puesto que los campos más relevantes de esta información han sido tenidos en cuenta en la estructura unificada). Al igual, el tener esta información conjunta y tratarla como si proviniera de una sola fuente, nos permite ordenar los productos por igual, independientemente del portal en los que estén ofertados.

Esta arquitectura está formada por tres niveles: El Nivel de Usuario o de Aplicación (correspondiente a la interfaz de usuario), el Nivel de Integración (nivel donde se produce la mayor parte del procesamiento de la información) y el Nivel de Fuente De Datos (donde se produce la funcionalidad de extracción de información de las diversas fuentes). En la Figura 7 se observa gráficamente el esquema de esta arquitectura.

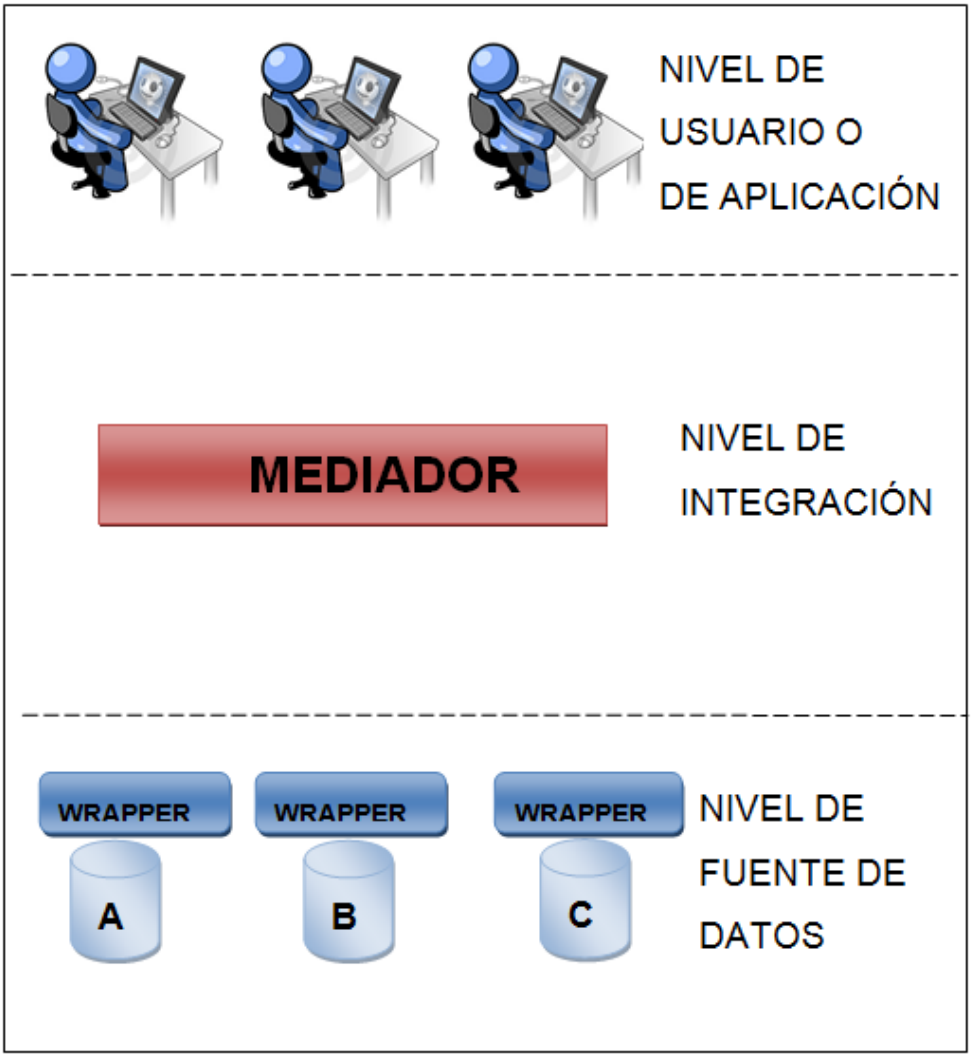

Figura 7: Arquitectura Mediador

4.1.2. Características de la arquitectura de Mediador.

Las características de la arquitectura de Mediador son las siguientes:

- **Transparencia de información:** El usuario realizará una consulta a los distintos portales que contempla la aplicación. Así esta consulta se realiza de forma transparente para el usuario, ya que el objetivo es que el usuario no perciba que está consultando en diferentes servidores, él tiene que tener la sensación de consultar solo uno. Esta función de transparencia y ocultación de información no relevante, se realiza en el Nivel de Integración que se encarga de recoger la información proveniente del Nivel de Fuente De Datos, procesarla, integrarla y devolverla al usuario un formato único y homogéneo.

- **Flexibilidad:** La arquitectura Mediador se caracteriza por ser una arquitectura muy flexible al permitir la extracción e integración de datos de cualquier tipo de fuente independientemente del tipo de esta (servidores web, bases de datos, documentos…). Cada wrapper es específico para una fuente de datos y se implementará de forma distinta dependiendo del tipo de fuente a la que este asociada. Esta flexibilidad es muy beneficiosa para el caso en el que en un futuro se plantee la inclusión de fuentes que no provengan del mismo formato que las actuales.

- **Escalabilidad:** La arquitectura Mediador también se caracteriza por ser una arquitectura escalable al permitir de una forma sencilla la inclusión de nuevas fuentes de búsqueda. Esto se realizaría mediante la inclusión de un nuevo wrapper en el Nivel de Fuente de Datos.

- **Personalizable:** Esta arquitectura nos permite realizar los ajustes necesarios en las distintas capas para ajustarla a los diferentes objetivos de las distintas aplicaciones. De esta manera podemos implementar transformadores y modificadores de orden de dicha información, dando a la aplicación un poder selectivo de la información mucho mayor. El objetivo no es solo conseguir un acceso unificado a todos los datos extraídos, sino además ser selectivo con los mismos devolviendo los más relevantes para el usuario.

4.2. Niveles de la arquitectura

En los siguientes apartados explicaremos, con algo más de detalle que hicimos en puntos anteriores del presente documento, los 3 niveles de los que se compone la Arquitectura de Mediador. Estos niveles se diferencian por su función específica que cumplen en la arquitectura. La arquitectura mediador no se podría entender sin la presencia de ninguno de ellos, estos son: Nivel de Usuario o de Aplicación, Nivel de Integración o Nivel de Fuente de Datos. De cada uno expondremos sus características más importantes así como la explicación de su comportamiento y de su función real en la arquitectura final que hemos adaptado.

También explicaremos como se ha ido adaptando cada nivel de la arquitectura al proyecto que nos ocupa, viendo de forma concreta para que hemos utilizado y de que nos ha servido cada nivel en la implementación de la aplicación desarrollada.

4.2.1. Funcionamiento de cada nivel

En este apartado explicaremos con detalle como es el funcionamiento interno de cada uno de los 3 niveles.

4.2.1.1. Funcionamiento del Nivel de Usuario o de Aplicación

El Nivel de Usuario o Aplicación, es el nivel más alto de los tres (el que se encuentra más próximo al usuario). Está formado por la interfaz con la que se realiza la interacción de los usuarios finales con la aplicación. Se encarga de mostrar la información que va a llegar a los usuarios y de recoger los datos que estos quieren introducir en el sistema para realizar las búsquedas.

El funcionamiento está dividido en dos, según se refiera a la introducción de la información por parte del usuario o a la muestra de la misma por parte del sistema.

Introducción de información

En cuanto a la introducción de información por parte del usuario, la interfaz de la aplicación cuenta distintos mecanismos para realizar este proceso. El primero de ellos es escribiendo la cadena de texto a buscar manualmente mediante el uso de una caja de texto, es lo que llamamos búsqueda genérica. El segundo es seleccionando las características de un producto específico mediante un menú desplegable (es lo que llamamos búsqueda específica). Esta información se le pasará al nivel inferior (El Nivel de Integración). Junto con este texto también se seleccionan en este nivel los portales donde el usuario desea realizar la búsqueda.

Muestra de resultados

El otro tipo de funcionamiento es el de mostrar la información recibida de las búsquedas. En este caso el Nivel de Usuario o Aplicación recibe del Nivel de Integración la información de los resultados obtenidos. Esta información está en formato unificado y lo único que realiza el Nivel de Usuario es mostrarla, además de cargar las fotos de los productos de Internet (ya que el Nivel de Usuario recibe el link del Nivel de Integración con la dirección web donde se encuentra alojada la fotografía). Desde este nivel también se pueden elegir las diversas opciones de orden de los productos mostrados, aunque su procesamiento será llevado a cabo en el nivel de integración.

4.2.1.2. Funcionamiento del Nivel de Integración

El Nivel de Integración es el nivel que lleva casi todo el peso de procesamiento de la información en la arquitectura. Se puede decir que es el nivel más complejo en

cuanto a implementación ya que se encarga de transformar la información recibida del usuario en información específica para cada wrapper y pasarla al Nivel de Fuente de Datos. También se encarga de recibir los resultados de búsqueda de cada uno de los wrappers de este nivel y reunificarla, para que pueda ser recibida como si proviniera de una sola fuente por el Nivel de Usuario. Por el Nivel de Integración pasan todas las peticiones realizadas en la aplicación.

Su funcionamiento se divide en tres tareas. Al igual que el Nivel de Usuario, dependiendo de si el flujo que le llega es de entrada o de salida de información. Además se añade una función adicional que es la implementación del procesamiento del orden de los productos, así como la de la gestión de errores de conexión.

Peticiones recibidas desde el Nivel de Usuario

Por una parte tendremos el caso de que se reciba la petición de búsqueda del Nivel de Usuario. El Nivel de Integración recibirá la información del usuario, ya sea en formato textual o mediante una serie de opciones marcadas, y la transformará en un formato textual específico para cada wrapper al que se quiera llamar. Esta transformación se realiza basada en cómo cada portal asociado a un wrapper tenga estructurada su información. El objetivo principal es obtener los resultados más relevantes de cada uno de ellos, y como cada buscador interno de cada portal es diferente al igual que la forma de nombrar a los productos, es necesaria esta transformación del texto de búsqueda en un texto específico para cada uno de ellos. Finalmente esta información es mandado al Nivel de Fuentes De Datos donde se encuentran los wrappers.

Resultados recibidos del Nivel de Fuente de Datos

Los wrappers terminan de obtener los resultados de la búsqueda y transmiten estos resultados en el formato específico de cada uno al Nivel de Integración. Una vez recibidos estos datos se produce la reunificación de los mismos. El objetivo de esta reunificación es poder tener los campos comunes en cada registro de resultados, para poder ordenarlos y sobre todo presentarlos al usuario de forma homogénea. Estos datos se pasarán de manera conjunta al Nivel de Aplicación para que se muestren al usuario.

Peticiones de orden desde el Nivel de Usuario

La última tarea que se realiza en este nivel, es el procesamiento de la ordenación de los productos mostrados al usuario. El usuario podrá elegir entre mostrar los productos por un determinado orden, este orden puede ser:

- **Productos ordenados por precio:** hay dos tipos de orden: ascendente, en el cual en primer lugar de la lista de resultados aparecerán los productos con mayor precio hasta llegar al producto más barato que se colocará en último lugar, y descendente, el orden inverso al anterior.
- **Productos ordenados por relevancia:** hay dos tipos de orden: ascendente, en el cual en primer lugar de la lista de resultados aparecerán los productos cuyo nombre tenga más palabras coincidentes con el texto de búsqueda, dejando para último lugar los productos que tengan menos palabras coincidentes con el mismo, y descendente, el orden inverso al anterior.
- **Productos ordenados por portales:** en este orden los productos aparecerán ordenados por los portales en los que han sido encontrados. Así los productos de un mismo portal aparecerán seguidos los unos de los otros.

Una vez recibida la petición de orden desde el nivel superior, se procesará esta operación en el Nivel de Integración y se devolverán los productos otra vez, pero con el nuevo orden, al Nivel de Usuario.

4.2.1.3. Funcionamiento del Nivel de Fuente de Datos

El nivel de Fuente de Datos es el nivel más bajo que tenemos en la arquitectura. En él se encuentran los wrappers, que son los encargados de obtener la información de los diferentes portales Web de venta de productos tecnológicos. Su funcionamiento se divide en 2 dependiendo del flujo de la información:

Peticiones de búsqueda recibidas del Nivel de Integración

Para el flujo de entrada de información, al Nivel de Fuente de Datos le llega la información de peticiones de búsqueda desde el Nivel de Integración. En este nivel se navega hacia la página del portal, se selecciona la caja de búsqueda del mismo y se realiza la búsqueda de la información obtenida.

Extracción de los resultados de las búsquedas

Una vez hecha esta búsqueda entra en juego el otro modo de funcionamiento, que se refiere al flujo de salida. Cada wrapper extrae la información de los resultados devueltos por su portal asociado. Una vez extraídos los resultados de estos portales, son enviados al nivel superior, el Nivel de Integración.

4.2.2. Componentes de la aplicación

En este apartado veremos los componentes que forman cada nivel de la arquitectura. No se explicará el funcionamiento de los mismos en detalle ya que es el mismo funcionamiento de la arquitectura explicado en los apartados anteriores, pero si haremos hincapié en explicar las funciones de las que se encargan.

Nivel de Usuario

El Nivel de Usuario o de Aplicación está compuesto por una interfaz en la que el usuario puede interactuar con la aplicación realizando peticiones de búsqueda y recibiendo información de la misma.

Esta interfaz está especializada para sus distintas funcionalidades, en cuanto a la entrada de información, la interfaz cuenta con una caja de texto para realizar búsquedas genéricas indicando un texto cualquiera. Para realizar búsquedas especificas cuenta con menús desplegables para que el usuario pueda seleccionar las determinadas características de cada tipo de producto que quiera buscar. Para la función de representación de resultados se cuenta con una interfaz donde se muestran por filas los campos más relevantes de los mismos, como el nombre, el precio y la compañía. Así, para facilitar más la labor a los usuarios se les presenta una fotografía de cada producto y un link al mismo para que solo haciendo clic puedan adquirir el producto en su respectiva tienda. Además se permite ordenarlos por precio, relevancia y por portales. Se explicará el diseño de esta interfaz en detalle en el apartado 4.3.

Nivel de Integración

El Nivel de Integración está compuesto por una entidad llamada Mediador El Mediador es una pieza básica en la arquitectura ya que realiza todas las funciones del Nivel de Integración que hemos mencionado en el punto anterior. Así el Mediador recibe las peticiones de los usuarios y realiza una función de adaptación de las mismas para cada uno de los diferentes wrappers existentes, para que se pueda hacer una búsqueda específica para cada portal requerido.

Al Mediador también les llega la información extraída por el nivel inferior, Con lo cual, es el encargado de juntar esta información y enviarla al nivel superior y que el usuario pueda recibir toda la información en un formato único y homogéneo.

Por último el Mediador también es el encargado de procesar los resultados y aplicarles los órdenes contemplados para mostrar la información más relevante para el usuario. Así como de gestionar los errores producidos por la conexión.

Nivel de Fuente de Datos

En cuanto al Nivel de Fuente de Datos, está formado por entidades llamadas wrappers (Contenedores). Un wrapper es una entidad que está asociada a un sitio web y que extrae información del mismo. Estas entidades se encargan de realizar las búsquedas en cada portal con la información específica que le ha llegado del Nivel de Integración y extraer los resultados obtenidos de la misma, almacenarlos y enviarlos al nivel superior, donde será procesada por el Mediador.

Cada wrapper es específico para cada sitio web, con lo cual es muy costoso en términos de implementación ya que cada portal está estructurado de forma diferente, en unos la información está muy estructurada y en otros muy poco estructurada. Con lo cual implementar un wrapper para cada uno de ellos es una tarea bastante difícil de llevar a cabo, al tener que salvar diferentes dificultades sobre todo en las webs con información poco estructurada.

Estos wrappers utilizan la tecnología Selenium junto con el lenguaje de consulta XPath para abrir la web deseada, acceder a sus elementos con el objetivo de realizar la búsqueda en el servidor y extraer la información de los productos una vez obtenidos los resultados de la búsqueda.

4.3. Diseño de la interfaz

El objetivo planteado en cuanto se comenzó a acometer el diseño de la interfaz es que fuera un diseño simple e intuitivo para el usuario. Para ello lo primero que se hizo fue revisar las aplicaciones existentes y estudiar su interfaz, con el objetivo de conocer las interfaces actuales que estaban utilizando las aplicaciones similares a la nuestra. Durante esta observación se fue tomando nota de la disposición de elementos y organización de estos en las diferentes interfaces, extrayendo lo que se pensó que era positivo y que podría ser válido para aplicarlo a la aplicación, y descartando lo que se pensó irrelevante.

Una vez que se realizó este estudio, se pasó a agrupar todas las funcionalidades de la aplicación que se correspondían con áreas o distintas interfaces a implementar. Estas áreas se pueden dividir en cuatro y van a explicarse en los siguientes apartados, en los que están expuestas por orden lógico (orden en el que el usuario interacciona con las mismas para llevar a cabo la búsqueda).

4.3.1. Pestañas

Así, para organizar las diferentes funcionalidades de la aplicación se decidió realizar pestañas para cada una de ellas, tanto para las diferentes categorías de la

búsqueda específica, como para la búsqueda genérica, la muestra de resultados y las opciones de la aplicación.

La elección de implementar pestañas, se debió a que actualmente los usuarios están acostumbrados a utilizarlas en los navegadores web. Todos los navegadores web más comunes poseen navegación por pestañas, con lo cual este tipo de navegación es utilizado por los usuarios de forma asidua, así que están acostumbrados a ellas. Además posee la capacidad que de un simple vistazo rápido desde cualquier pantalla, se puede visualizar todos los tipos de búsqueda, simplemente observando el título de cada pestaña. Así se convierte en una manera efectiva de implementar esta funcionalidad. Las distintas pestañas contempladas son las siguientes:

- **General:** se corresponde con la búsqueda genérica.
- **Televisiones:** se corresponde con la búsqueda específica de monitores y televisiones.
- **Fotografía-MP3:** se corresponde con la interfaz de búsqueda específica de cámaras fotográficas y de reproductores MP3.
- **Memoria:** se corresponde con la búsqueda específica de tarjeta de memoria y almacenamiento USB.
- **Informática:** en ella se encuentra la interfaz correspondiente con la búsqueda específica de ordenadores de sobremesa, portátiles e impresoras.
- **Resultados:** se corresponde con la interfaz de muestra de resultados.
- **Opciones:** se corresponde con la interfaz de muestra de las opciones de configuración de la aplicación.

En la Figura 8, podemos las distintas pestañas de nuestra aplicación:

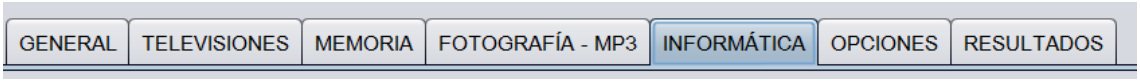

Figura 8: Pestañas de la aplicación

4.3.2. Definición de la búsqueda

La definición de la búsqueda se realiza de forma distinta en la búsqueda genérica y en la búsqueda específica. Así en la búsqueda genérica simplemente te introducirá un texto mediante una caja de búsqueda y en la búsqueda específica se seleccionaran las características de un producto mediante unas listas desplegables. El diseño de la interfaz por lo tanto será distinto para cada funcionalidad.

Para la búsqueda genérica, se ha implementado una caja de texto grande y en el centro de la interfaz, para que el usuario introduzca el texto deseado. El relevante tamaño de la caja de texto en comparación con el resto de la interfaz es para enfatizar el tipo de búsqueda y la importancia que tiene el proceso de introducción del texto. Así, el estar centrada, se basa en el diseño que tiene Google en su buscador y la razón es porque los usuarios están acostumbrados a este buscador y a que la caja de búsqueda esté en esta posición de la interfaz.

Además se ha incluido un texto explicativo que acompaña a la caja indicando la acción que tiene que realizar el usuario, junto con el número 1, que se interpreta como el primer paso que el usuario tiene que realizar para empezar la búsqueda. Este paso es la introducción de texto.

En la Figura 9 se puede observar la caja de texto implementada para la búsqueda genérica:

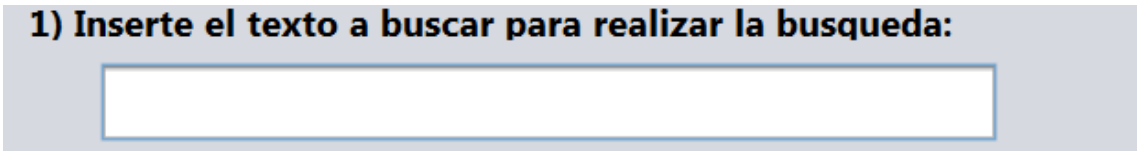

Figura 9: Caja de texto correspondiente a la búsqueda genérica

En cuanto a la búsqueda específica, la selección de características, como hemos dicho, se realizar mediante listas desplegables. Como estas características son diferentes para cada tipo de productos, la interfaz se ha separado por estos tipos, mostrando de cada uno sus características en las listas desplegables. Estas listas vienen acompañadas de un texto explicativo que define la característica que se está eligiendo.

Una vez seleccionado una característica, se mostrará marcada en la lista, indicando que se encuentra activa para realizar la búsqueda basándose en ella. También se ha incluido una casilla (radio button) para seleccionar el tipo de producto a tratar. Así una vez que se selecciona un tipo de producto, sus listas desplegables quedan activadas, desactivando todas las opciones del resto de productos contemplados en la misma pantalla. Esta activación/desactivación de las listas desplegables

hace más intuitiva la aplicación, ya que el usuario observa de simple vistazo el tipo de producto que se encuentra activo actualmente.

Además, al igual que en la búsqueda genérica, se ha incluido un texto explicativo que acompaña a la caja indicando la acción que tiene que realizar el usuario, junto con el número 1, que se interpreta como el primer paso que el usuario tiene que realizar para empezar la búsqueda. Este paso es la selección del tipo de producto y sus características.

Un ejemplo de la interfaz de búsqueda específica, se puede observar en la Figura 10, que corresponde a los productos enmarcados en la categoría de "Informática":

Figura 10: Productos de la categoría "Informática"

4.3.3. Selección de portales

La tercera funcionalidad es la selección de los portales donde se quiere buscar. Esta funcionalidad se ha implementado mediante la selección de casillas.

Como en las demás funcionalidades, se ha incluido un texto explicativo que acompaña a esta área de la interfaz, indicando la acción que tiene que realizar el usuario, junto con el número 2, que se interpreta como el segundo paso que el usuario tiene que realizar para empezar la búsqueda. Este paso es la selección de portales donde se quiere buscar.

La interfaz de esta funcionalidad tiene el mismo diseño para todos los tipos de búsqueda, al ser una funcionalidad común a todos ellos. En la Figura 11, se puede observar la interfaz de selección de tiendas:

Figura 11: Interfaz de selección de tiendas

4.3.4. Muestra de resultados

Una de las interfaces en las que el diseño es una parte crucial es la de la muestra de resultados. En esta funcionalidad, hay que tener en cuenta muchos aspectos que bien contemplados pueden dar al usuario unas ventajas que no tendría si el diseño de los mismos fuera pésimo.

Así, vamos a proceder a explicar cada uno de estos aspectos a tener en cuenta en esta interfaz.

4.3.4.1. Número de productos a mostrar y navegación

El número elegido fue finalmente de 20 productos a mostrar por cada página. En total se contemplan 5 posibles páginas de resultados, con un total de 100 productos, que son el máximo que puede devolver una búsqueda, ya que en las opciones de configuración no se deja que el usuario elija una búsqueda de más de 100 productos.

Además, debido al diseño de la presentación de cada producto, en la interfaz solo se muestran 5 productos a la vez, pudiendo navegar por los demás a través de un scroll. Esto se ha decidido así, para dar énfasis a los 5 productos más relevantes en cada búsqueda, que son los más importantes para el usuario. Siempre dando la posibilidad de una forma sencilla e intuitiva de ver el resto de productos.

Las demás páginas se pueden observar de manera sencilla mediante los botones de navegación, situados en la esquina superior derecha de la interfaz. La colocación en esta parte de la interfaz es para mantener la similitud con la colocación de los mismos en las diferentes páginas web donde se realizan operaciones de este tipo. Así, el usuario está acostumbrado a esta ubicación.

En la Figura 12 se puede observar la disposición de los productos en la pantalla:

Figura 12: Muestra de resultados en la aplicación

4.3.4.2. Atributos a mostrar en los productos

Un aspecto crucial para un buen diseño de la interfaz es la colocación y el número de atributos a mostrar de un producto. El objetivo de este diseño es que le usuario tenga la máxima información posible sobre los mismos teniendo que leer lo mínimo de la interfaz.

Así, finalmente los atributos recogidos son los siguientes por orden de izquierda a derecha en la interfaz:

- **Fotografía:** una fotografía en tamaño reducido del producto para que el usuario pueda ver de qué se trata y no obtenga ninguna confusión. (Como por ejemplo que distinga si el producto es un iPhone o una funda para iPhone, cuyo nombre puede ser muy parecido).
- **Portal de venta:** logotipo del portal donde se vende dicho producto. Se ha incluido para que el usuario sepa de una manera rápida que portal es el que está ofertando el producto.
- **Nombre del producto:** nombre con el que el portal describe el producto. Este nombre se ha colocado en la parte central y se le ha dado una importancia máxima, ya que es lo más descriptivo que se ofrece del producto. En él va incrustado el enlace de acceso a la página de compra del producto.

- **Precio en euros**: el precio se muestra de una manera clara y sencilla en esta casilla, así primero se coloca el precio en euros ya que será común a todos los productos.
- **Precio en dólares:** en esta casilla se colocará el precio original en dólares del producto, en el caso de que la tienda venda los productos en dólares.

En la Figura 12 también se puede observar en detalle cómo se muestran los atributos de un producto determinado.

4.3.4.3. Orden de resultados

El diseño de esta funcionalidad es un diseño sencillo y muy común a la mayoría de las páginas webs que la ofrecen. Así se ha optado por implementarla mediante una lista desplegable de opciones, indicando los diferentes tipos de orden de manera explícita y textual, que al pulsar sobre el orden requerido, la operación se llevará a cabo.

Además se muestra un texto indicando el número de resultados encontrados, el número de páginas disponibles y el número de página actual. También se incluyen los botones de navegación entre las páginas “Siguiente” y “Anterior” a la derecha de la interfaz.

La mayoría de interfaces actuales similares, incluyen esta funcionalidad en la parte superior de la interfaz, con lo cual colocarlo aquí es lo más sensato teniendo en cuenta que la aplicación tiene que ser intuitiva para el usuario.

En la Figura 13 se observa como se ha implementado la funcionalidad del orden de los resultados.

Figura 13: Interfaz de orden de resultados

Finalmente en la Figura 14, se muestra la interfaz completa de muestra de resultados:

Figura 14: Interfaz de muestra de resultados

5. Comportamiento del sistema

Una vez que hemos explicado el análisis y el diseño del desarrollo de la aplicación de este proyecto, vamos a proceder a modelar el comportamiento de esta, con el objetivo de explicar cómo funciona el sistema. Así se contemplará el flujo normal de la ejecución del mismo.

Para ayudar a entender de forma más clara este comportamiento, utilizaremos diagramas de secuencia UML, para cada una de las funcionalidades que el sistema ofrece al usuario. De esta manera se podrá observar de forma gráfica y sencilla el comportamiento de cada una de estas funcionalidades.

Antes de pasar a explicar estas funcionalidades, tenemos que comentar que en los diagramas de secuencia se ha contemplado el flujo entre entidades en el sistema. Estas entidades se corresponden en la etapa de implementación con las clases de código de la aplicación. Como la implementación se explicará en el capítulo 6, vamos a resumir brevemente cada una de las entidades que aparecerán en estos casos de uso, para que el lector pueda obtener el completo conocimiento del comportamiento del sistema.

Las entidades aparecidas en estos diagramas, son las siguientes, agrupadas por el nivel de la arquitectura donde se encuentren:

- **Nivel de Usuario:**
  - **Entidad View:** es la entidad que representa la interfaz del sistema.
- **Nivel de Integración:**
  - **Entidad Mediator:** es la entidad que actúa como vehículo de peticiones y datos en el sistema. A esta entidad, le llegan estos datos o peticiones procedentes de las demás entidades y su función es redirigir estos datos o peticiones a las entidades adecuadas.
  - **Entidad Filter**: es la entidad encargada de procesar el orden de la lista de productos devuelta como resultado de la búsqueda.
  - **Entidad SpecialSearchString:** su función es la de crear las cadenas de texto específicas para cada uno de los wrappers, en el caso de la búsqueda genérica.
- **Nivel de Fuente de Datos:**
  - **Entidad Wrapper:** hay una entidad para cada uno de los distintos wrappers que están asociados a cada portal web. Su función es la de

realizar la búsqueda en estos portales, extraer la información de los resultados obtenidos y transferirla al nivel superior.

También comentar, que en los diagramas aparecerán los métodos reales de las clases finalmente implementadas, al igual que los datos transferidos entre entidades. Para entender estos diagramas no hace falta conocer cada uno de los métodos, ya que son meros activadores de las acciones a producirse y estas se explican con detalle en este capítulo, al igual ocurre con los datos transferidos. Para más información, se explicarán en el capítulo 6 en profundidad.

5.1. Búsquedas

5.1.1. Búsqueda genérica

En la Figura 15, procedemos a mostrar el diagrama de secuencia de una búsqueda genérica del usuario en la aplicación.

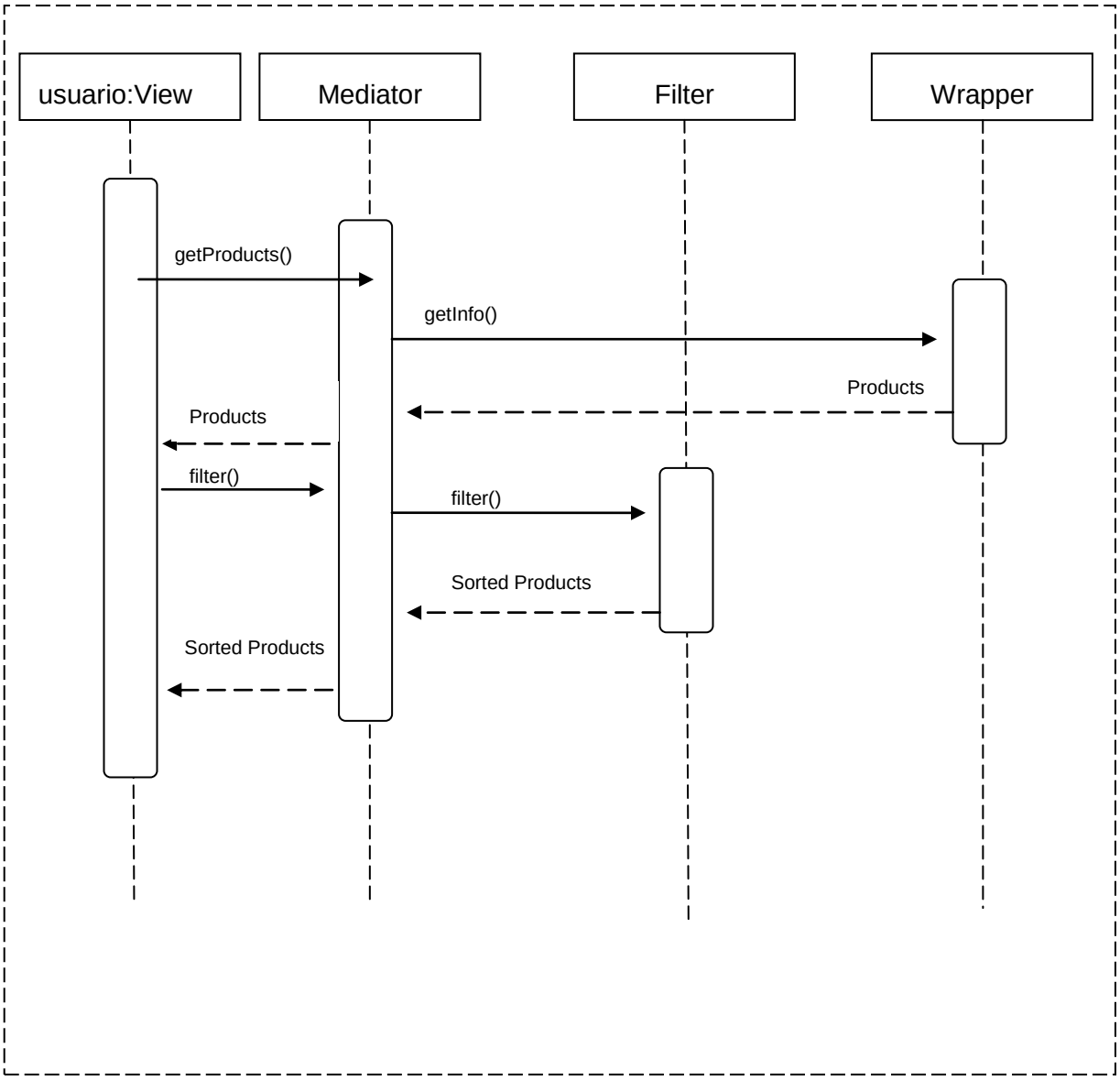

Figura 15: Diagrama de secuencia de la búsqueda genérica

Descripción del comportamiento de Búsqueda Genérica:

- El usuario desde la interfaz, entidad View, escribe el texto a buscar en la caja de texto de búsqueda genérica, selecciona los portales deseados y pulsa el botón “Buscar”. En ese momento la cadena de texto introducida es enviada a la entidad Mediator a través del método getProducts().

- En Mediator, esta petición se procesará y dependiendo de las opciones de portales marcadas donde se quiera buscar, se enviará la información específica para cada uno de ellos, a su respectivo wrapper. Esto se hará utilizando los métodos getInfo() de cada wrapper en particular.

- En los wrapper, se realizará una búsqueda en las páginas webs de los portales asociados (utilizando la tecnología Selenium y XPath) y se obtendrá una serie de resultados relevantes. Estos resultados son extraídos y almacenados en entidades de almacenamiento de datos para los productos. Finalmente se devuelven a Mediator.

- En Mediator, esta información es recogida, reunida y enviada a View de forma conjunta, donde serán expuestos al usuario.

- En View el usuario puede querer ordenar los productos en alguno de los órdenes soportados, así llamarán a la entidad Mediator para este cometido a través del método filter().

- La entidad Mediator a su vez transferirá la llamada a la entidad Filter donde se le aplicarán el orden seleccionado. Así esta información será ordenada con el objetivo de poder mostrar al usuario la información más relevante en cuanto al orden requerido por él. Esta información es devuelta de nuevo a Mediator.

- Mediator devuelve estos productos ordenados de nuevo a View donde serán expuestos al usuario como resultado de su búsqueda.

5.1.2. Búsqueda específica

Las búsquedas específicas son similares a las genéricas pero con una pequeña peculiaridad que comentaremos. De igual manera, aunque el procedimiento es bastante similar, procedemos a explicar sus particularidades. El diagrama queda reflejado en la Figura 16.

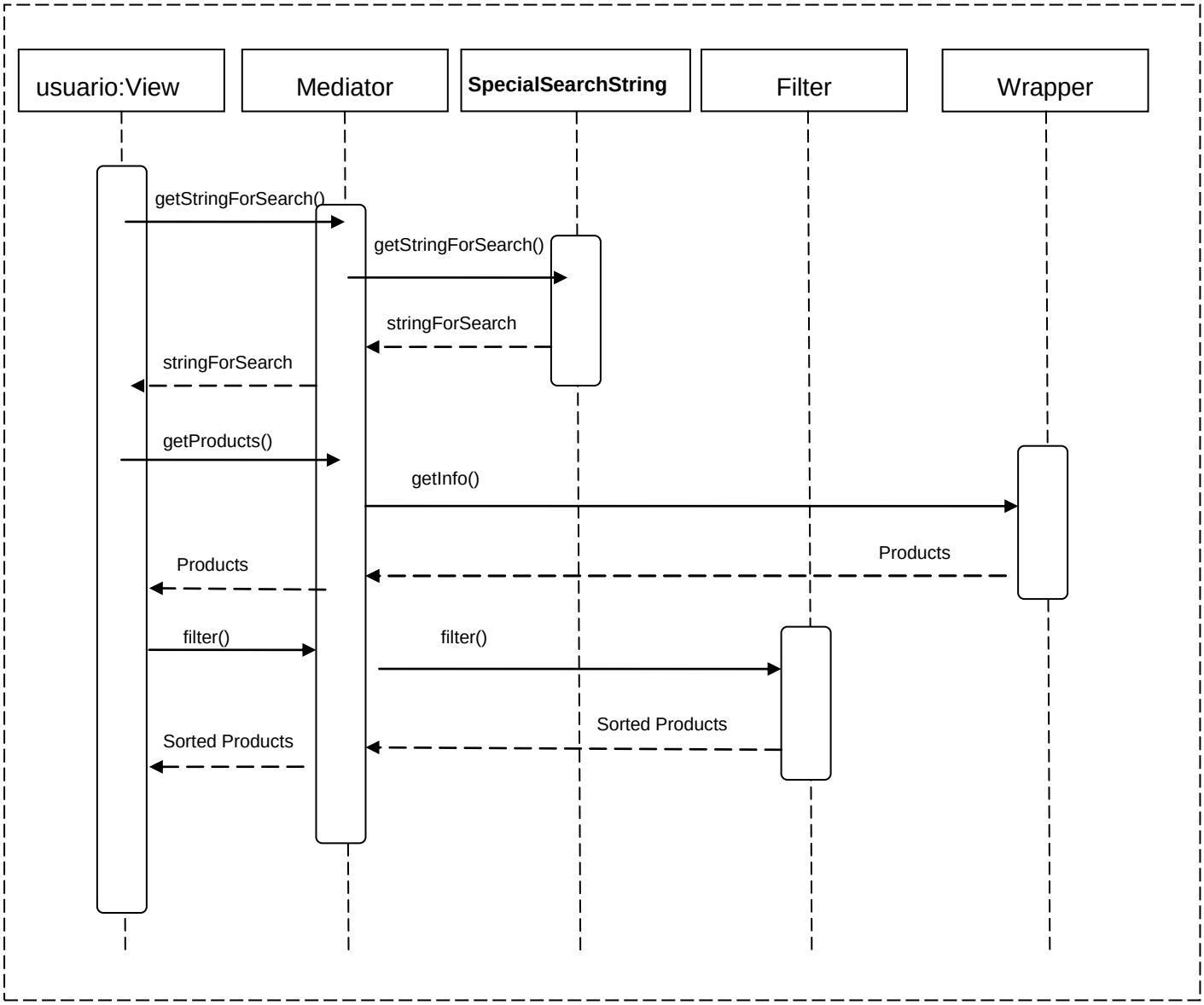

Figura 16: Diagrama de secuencia de la búsqueda específica

Descripción del comportamiento de búsqueda específica:

El comportamiento de la búsqueda específica difiere en el de la búsqueda genérica en varios aspectos que procedemos a comentar.

En la pestaña seleccionada por el usuario, el usuario marca el tipo de producto que quiere buscar y selecciona las características del mismo a través de las listas desplegables que existen para cada tipo. Una vez seleccionadas, el usuario elige los portales en los que desea realizar la búsqueda y finalmente pulsa el botón “Buscar”, enviando así la petición de búsqueda.

En Mediator, esta petición se procesará y se enviará a la entidad SpecialSearchString para obtener las cadenas de texto finales que se utilizarán en cada búsqueda.

SpecialSearcthString recibe las características de productos seleccionadas por el usuario desde Mediator y genera dependiendo de las opciones de portales marcadas y de estas características, las cadenas finales. Una vez creadas las devuelve de nuevo a Mediator como resultado de su petición. Mediator devuelve las cadenas de texto a la entidad View, que inmediatamente realiza la petición final de búsqueda, utilizando el método getProducts() . El resto del proceso es el mismo que en el caso de la búsqueda genérica.

5.2. Resultados (exposición y operaciones)

En la pantalla de exposición de resultados, la aplicación permite realizar diversas operaciones, como navegar entre páginas, u ordenar resultados según algún criterio. En este apartado explicaremos el comportamiento de la aplicación al realizar estas operaciones mencionadas.

5.2.1. Ordenación de resultados

La aplicación permite a los usuarios realizar una ordenación de resultados dependiendo del precio de los mismos y de su relevancia. Así, hay tres opciones de orden:

- Por relevancia (orden ascendente y descendente).
- Por precio (orden ascendente y descendente).
- Por agrupación de portales.

El orden por precio consiste en mostrar la lista de resultados ordenada por productos de mayor a menor precio o viceversa dependiendo del sentido del orden escogido. Mientras que el orden por portales consiste en mostrar la lista de resultados ordenada por productos que pertenezcan al mismo portal, así aparecerán los productos dispuestos agrupados por portales en los que han sido encontrados. Explicaremos con más detalle cómo funciona el orden por relevancia, ya que es un poco más complejo que los anteriores.

Al realizar la búsqueda utilizando los buscadores internos de cada fuente, la información ya pasa por unos funciones que realizan un ranking de relevancia, que están implementados en cada buscador de cada portal. Aunque estos rankings no serán los mismos para cada fuente (es decir los de Amazon no tienen por qué ser los mismos que los de FocalPrice), en general siguen patrones muy similares y garantizan una alta relevancia en las búsquedas, que es lo que nos interesa. Aun así hemos incluido una mejora en la tarea de obtener los productos más relevantes.

Esta mejora, se realiza una vez que los productos han sido mostrados en la interfaz. Se trata de una manera de ordenar estos productos basándose en la relevancia de estos respecto al texto introducido por el usuario.

El funcionamiento de este orden es el siguiente: se compara las palabras introducidas por el usuario en la búsqueda con las palabras contenidas en los títulos de los productos mostrados. Así los productos se ordenarán de mayor número de palabras coincidentes de su título con el texto de búsqueda a menor número de ellas.

Un ejemplo es el siguiente: si el usuario introduce como búsqueda el texto “iphone 3g 16GB” y el primer resultado mostrado en la búsqueda se titula “Apple iphone 16GB 3gs Black” y el segundo “Apple iphone 16GB 3g”, al coincidir mayor número de palabras el título del segundo que el del primero con el texto de búsqueda, se intercambiaría el orden siendo más relevante el segundo producto y pasando a la primera posición.

También existe la posibilidad de realizar el orden inverso, ordenando los productos de menor a mayor relevancia.

El diagrama de secuencia para entender el comportamiento de dicha operación viene definido en las figuras de los diagramas anteriores, donde comienza con la llamada por parte de View al método filter() y termina cuando se recibe los productos ordenados:“Sorted Products”.

Descripción del comportamiento de ordenación de resultados:

- El usuario una vez realizada la búsqueda observa los resultados obtenidos en la interfaz de muestra de resultados. En esta interfaz el usuario podrá elegir entre ordenarlos por uno de los criterios posibles. Para ello activa el botón correspondiente y la entidad View llamará a la entidad Mediator para este cometido a través del método Filter().

- La entidad Mediator a su vez transferirá la llamada a la entidad Filter donde se le aplicarán el orden seleccionado. Así esta información será ordenada con el objetivo de poder mostrar al usuario la información más relevante en cuanto al orden seleccionado. Esta información es devuelta de nuevo a Mediator.

- Mediator devuelve estos productos ordenados a View donde serán expuestos al usuario como resultado de su búsqueda.

Este comportamiento es común para todos tipos de orden, con lo cual no se considera relevante explicar los comportamientos para cada orden en particular.

5.2.2. Navegar por los resultados

Cuando los resultados de la búsqueda son numerosos y no se pueden mostrar en una sola página de la interfaz de muestra de resultados, estos se organizarán en varias páginas. Así, el usuario podrá navegar entre las distintas páginas para ver al completo todos los resultados obtenidos en la búsqueda. Procedemos pues, a mostrar el diagrama de secuencia de esta operación en la Figura 17.

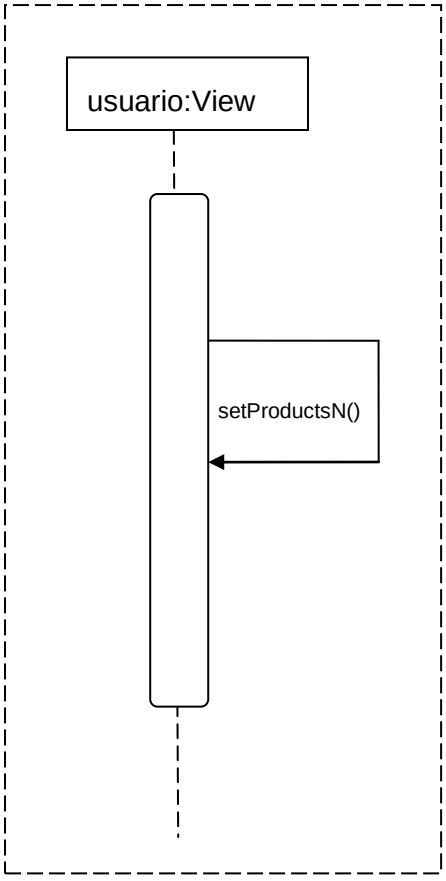

Figura 17: Diagrama de secuencia de la navegación por los resultados

Descripción del comportamiento de la navegación por los resultados:

- El usuario una vez realizada la búsqueda, observa los resultados obtenidos en la pantalla de muestra de resultados. En esta pantalla, si los resultados son numerosos, se agruparán en páginas, por las que el usuario podrá navegar. Así los botones de navegación se activarán automáticamente, indicando que hay más páginas de resultados disponibles.
- Cuando el usuario pulsa uno de los botones de navegación, la entidad View, llamará a un método contenido en ella que será quien implemente la funciona de navegar a la página seleccionada. Permitiendo al usuario desplazarse entre las distintas páginas que contienen el resto de resultados de la búsqueda.
- Los nuevos resultados serán mostrados al usuario en la misma pantalla de presentación de resultados.

6. Implementación del sistema

Una vez visto el comportamiento del sistema, procederemos a explicar la implementación del mismo. Primero veremos el diagrama de clases de la aplicación, para obtener una vista general de toda la implementación del sistema. Después, veremos en profundidad las clases que se han implementado en cada capa, explicando cada una de ellas, así como también explicar cómo se ha ido desarrollando cada una de las capas.

Finalmente explicaremos esta implementación del sistema en función de sus características de escalabilidad, disponibilidad, robustez, reutilización, mantenimiento y portabilidad.

En el Anexo B de esta memoria se ofrece una descripción de la implementación del código de las principales funcionalidades de la aplicación, así se puede ver el uso de las distintas tecnologías en la misma.

6.1. Diagrama de clases:

En la Figura 18 se puede observar el Diagrama de clases de la aplicación. En él, se puede ver las distintas clases que se han implementado y las relaciones entre ellas. Así en total se han implementado 14 clases, divididas entre los distintos niveles de la arquitectura de la siguiente forma:

- Nivel de Usuario o aplicación:
  - Clase View
- Nivel de Integración
  - Mediator
  - Product
  - Filter
  - SpecialSearchString
  - SearchTimer
- Nivel de Fuente de Datos
  - Ebaywrapper
  - Amazonwrapper
  - Optizewrapper
  - PCOnlinewrapper

- Extremewrapper
- Focalwrapper
- Dynoswrapper
- Exchangewrapper.

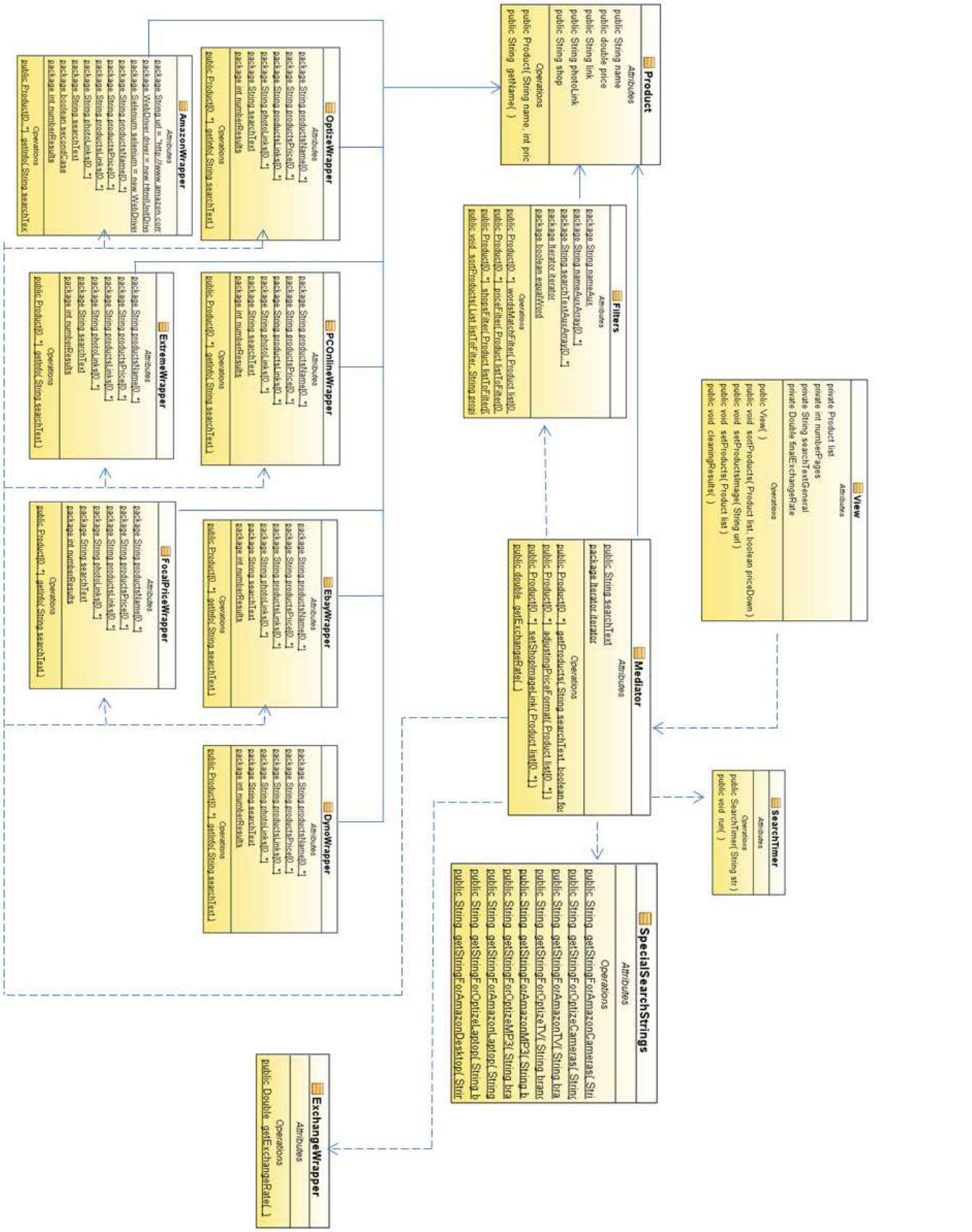

Figura 18: Diagrama de clases

6.2.1. Implementación del Nivel de Usuario o Aplicación

El objetivo principal a la hora de implementar la interfaz era conseguir una aplicación con una imagen sencilla que fuera intuitiva para el usuario y nada pesada en cuanto a consumo de recursos.

Tras barajar varias posibilidades a la hora de implementar la interfaz, finalmente se optó por utilizar las clases disponibles en las librerías Swing y AWT de Java.

Esta elección también se basó en la sencillez que suponía al utilizar el entorno de programación utilizado, Netbeans. Este entorno, trae incorporada una herramienta de diseño de interfaces que utilizan estas librerías, con lo cual el diseño a priori iba a resultar algo sencillo de realizar. En la Figura 19, se observa el diseño de la interfaz con la herramienta proporcionada por Netbeans.

Figura 19: Diseño de la interfaz en Netbeans

Con lo cual, al tener integrado en el mismo entorno elementos visuales y código, fue sencillo aplicar el comportamiento a los controles existentes en la interfaz, estos controles son los botones, pestañas y check boxes. Pues el nexo que enlaza estos elementos con su código es muy sencillo en Netbeans.

6.2.1.1. Clase View

En cuanto a las clases, el Nivel de Usuario o de Aplicación está formado por una única clase que se comunica con el Nivel de Integración. Esta clase llamada "View", se puede ver representada en la Figura 20.

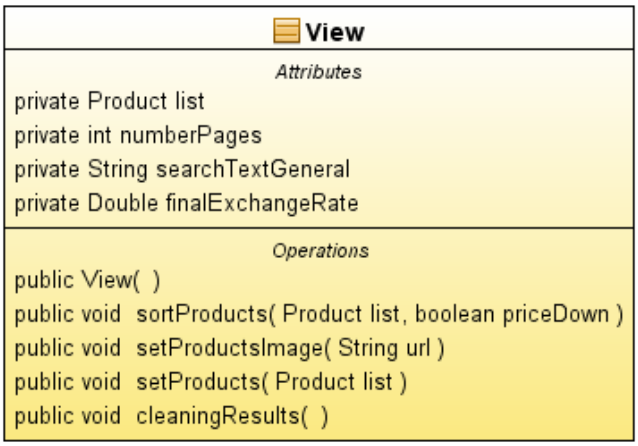

Figura 20: Clase View

Esta clase eta compuesta por 4 atributos y 5 métodos, de los cuales pasamos a ofrecer su descripción a continuación.

6.2.1.1.1. Atributos de la clase View

- **list:** lista de productos que se utiliza para guardar los productos obtenidos como resultado de la búsqueda,
- **searchTextGeneral:** variable donde se almacena el texto de búsqueda introducido por el usuario (en el caso de la búsqueda genérica) o formado por la aplicación tras seleccionar las características de la misma (en el caso de la búsqueda especifica).
- **finalExchangeRate:** atributo utilizado para guardar el tipo de cambio de moneda actual entre dólares y euros.
- **numberPages:** atributo utilizado para contabilizar el número de páginas de resultados existentes.

6.2.1.1.2. Metodos de la clase View

- **view():** Constructor de la clase.
- **sortProducts():** ordena los productos considerando el criterio indicado por el usuario en la interfaz.
- **setProductImage():** carga las imágenes de los productos obtenidos en la interfaz. Para ello utiliza las URL de las imágenes de cada producto, que son recibidas del nivel inferior.
- **setProducts():** carga los resultados obtenidos en forma de productos en la interfaz de muestra de resultados.
- **cleaningResults():** limpia los resultados en la pantalla de muestra de resultados.

6.2.1.1.3. Relaciones de la clase View

- **Clase Mediator:** La clase View se relaciona con la clase Mediator enviando los datos recibidos del usuario en la interfaz y recogiendo de esta clase los resultados obtenidos en la búsqueda. Así también le pide a esta, las cadenas de texto específicas para cada wrapper, convertidas como resultado de las características de los productos marcadas por el usuario en la búsqueda especifica.

6.2.1.2. Caché de imágenes

En las funcionalidades de orden y de navegación entre páginas de resultados se percibió una vez terminada la aplicación un rendimiento poco óptimo de las mismas. Esto se debía a que cada vez que se refrescaba la interfaz de resultados, la aplicación tenía que descargar de nuevo las imágenes de los productos, con el consiguiente consumo de recursos temporales que esto suponía.

Así, para solventar este problema, se creó un directorio caché de imágenes en el cual una vez devueltos los resultados de la búsqueda a la interfaz, se procedería a la descarga de las imágenes de los productos obtenidos y a su posterior almacenamiento en el mismo. Este directorio local es seleccionable por el usuario en las opciones de la interfaz.

Una vez almacenadas las imágenes, al realizar una operación de orden o navegación de páginas de los productos mostrados en la interfaz, las imágenes a mos-

trar serán tomadas de este directorio en lugar de descargarse de Internet. Esto produce una notable mejora del rendimiento de estas operaciones.

También se ha incluido una opción en la interfaz para que el usuario tenga la posibilidad de vaciar este directorio utilizado como caché de las imágenes.

6.2.2. Implementación del Nivel de Integración

La implementación del Nivel de Integración se ha realizado íntegramente en código Java. Esta capa es la que sirve de intermediadora entre los wrappers y la interfaz, aparte lleva todo el procesamiento de los datos obtenidos de los usuarios, como la transformación de cadenas de texto de búsqueda y orden de los resultados de las búsquedas. En la Figura 21 se muestran las clases que componen este nivel y una explicación en detalle de las mismas.

6.2.2.1. Clase Mediator

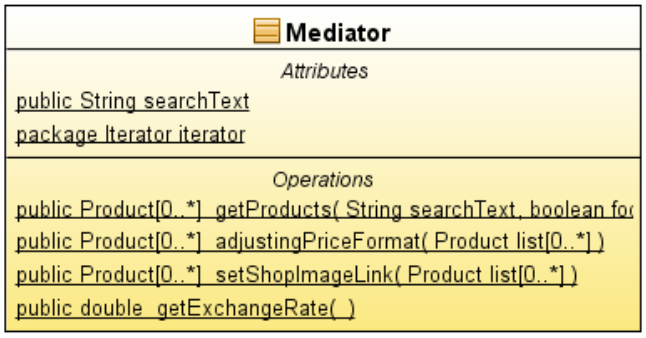

Figura 21: Clase Mediator

Es la clase que controla todas las comunicaciones de la aplicación Se encarga de transferir peticiones y datos al resto de las clases para que lleven a cabo las distintas funcionalidades de la aplicación. Procederemos a explicar sus atributos y métodos.

6.2.2.1.1. Atributos de la clase Mediator

- **iterator:** utilizado para iterar sobre los elementos de las listas de productos.

- **searchText:** texto de búsqueda introducido por el usuario recibido del Nivel de Usuario.

6.2.2.1.2. Métodos de la clase Mediator

- **getProducts():** es el método que llama a los diferentes wrappers con los datos pasados desde el Nivel de Usuario. Su función es recibir la petición de búsqueda con estos datos y obtener los resultados de la misma y devolverlos al nivel superior.
- **setShopImageLink():** obtiene la imagen de las distintas tiendas de productos tecnológicos de los que se ha realizado la búsqueda.
- **adjustinPriceFormat():** método que unifica el formato del precio de cara a la conversión de moneda que se va a realizar.
- **getExchangeRate():** método que realiza la conversión de precio de dólares a euros para los portales que lo requieran.

6.2.2.1.3. Relaciones de la clase Mediator

Esta clase se relaciona con todas las demás clases de la aplicación, ya que todos los datos y peticiones de la aplicación pasan a través de ella. Veamos las relaciones individualmente:

- **Clase View:** recibe datos de esta clase. Estos son los datos que son introducidos por el usuario. Así también envía a ella los resultados de la búsqueda obtenidos de los wrappers. Lo mismo ocurre con las cadenas de texto convertidas como resultado de la selección de opciones en la búsqueda especifica.
- **Clases de los wrappers:** Envía a los distintos wrappers las cadenas de texto introducida por el usuario (previamente transformadas), para que estos puedan realizar la búsqueda. También recibe de ellos, los datos de los productos obtenidos como resultado de las búsquedas realizadas. Además realiza la misma función con el wrapper encargado de obtener el tipo de conversión de moneda actual.
- **Clase Filter:** Envía a esta clase los productos obtenidos del Nivel de Usuario para que sean ordenados allí, recibiendo después los productos finales que han quedado ordenados y que serán enviados al nivel superior, para ser mostrados al usuario.

- **Clase SpecialSearchString:** Envía la petición a esta clase para obtener de ella las cadenas de texto convertidas como resultado de la selección de opciones en la búsqueda especifica.
- **Clase SearchTimer:** Crea una instancia de esta clase, cuando comienza la búsqueda, para gestionar la perdida de conexión.

6.2.2.2. Clase Filters

En la Figura 22, se observa los métodos y atributos de la clase Filters.

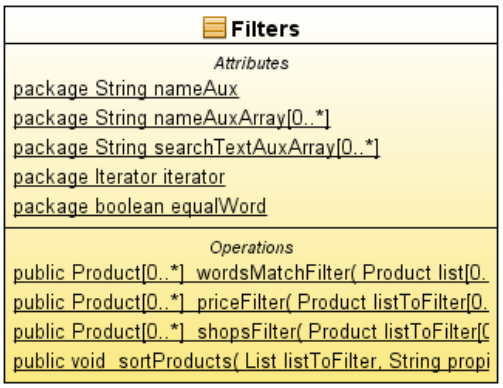

Figura 22: Clase Filters

En esta clase es donde se implementan los métodos que transforman el orden de los resultados obtenidos de la búsqueda, cuando el usuario quiere ordenarlos por algún criterio de los permitidos. Estos órdenes tienen que ver con la relevancia con el texto introducido, con los precios de los productos y con los portales donde se ofertan.

Así, el objetivo de estas funciones es mostrar los resultados más relevantes en el orden en el que el usuario requiera. Pasemos a ver los atributos y los métodos de esta clase.

6.2.2.2.1. Atributos de la clase Filters

- **nameAux:** variable auxiliar utilizada en la comparación de palabras.
- **nameAuxArray:** variable auxiliar utilizada en la comparación de palabras.
- **searchTextAuxArray:** variable auxiliar utilizada en la comparación de palabras.
- **equalWord:** variable auxiliar utilizada en la comparación de palabras.
- **iterator:** variable utilizada para iterar por los elementos de una lista.

6.2.2.2.2. Métodos de la clase Filters

- **wordsMatchFilter():** implementa la funcionalidad del orden de relevancia. Compara la cadena de texto introducida por el usuario con el nombre de cada producto y contabiliza las palabras coincidentes, guardando el número de ellas. Así ordena la lista de productos de mayor a menor número de palabras coincidentes con el texto de búsqueda. Implementando así la funcionalidad del orden de relevancia.
- **priceFilter():**implementa la funcionalidad del orden de precios. Dependiendo de los parámetros pasados, se realizará el orden de manera ascendente o descendente. Así se comprara los precios de cada artículo ordenando la lista por este criterio.
- **shopsFilter():** implementa la funcionalidad del orden por portales de los productos.
- **sortProducts():**procedimiento auxiliar que ordena la lista consideran el atributo pasado de sus elementos.

6.2.2.2.3. Relaciones de la clase Filters

- **Clase Mediator:** Esta clase se relaciona con la clase Mediator recibiendo la lista de todos los productos obtenidos en la búsqueda y enviándole de vuelta esta lista, pero con los productos ordenados por el criterio seleccionado.

- **Clase Product:** utiliza una estructura de esta clase para almacenar los productos a ordenar.

6.2.2.3. Clase Product

Esta clase es la que define la representación de los productos, define como se guardará la información de cada uno de ellos. Se trata de una clase estructural con muy poca lógica. En la Figura 23, se observa los métodos y atributos de esta clase.

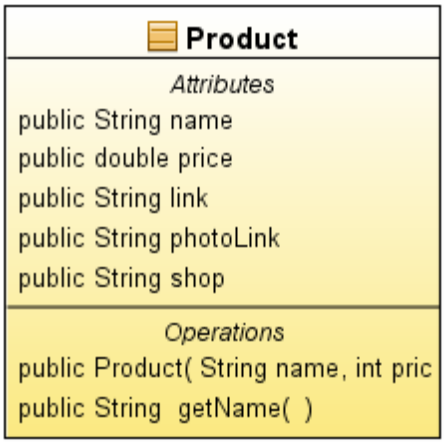

Figura 23: Clase Product

Está compuesta por los atributos que definen elementos de información de cada producto y por los métodos Set/Get para actualizar/obtener los valores de ellos. Explicaremos cada uno de estos atributos, pero solo daremos un ejemplo de métodos Set/Get ya que todos tienen la misma funcionalidad.

6.2.2.3.1. Atributos de la clase Product

- **name:** Nombre del producto.
- **link:** URL de la página de compra del producto en el portal donde es ofertado.
- **photolink:** URL de la imagen asociada al producto.
- **price:** precio del producto.
- **shop:** portal donde se ha encontrado el producto.

6.2.2.3.2. Métodos de la clase Product

- **Product():** constructor de la clase.
- **GetName():** obtiene el valor del atributo “name” del producto.
- **SetName():** establece el valor del atributo “name” del producto.

6.2.2.3.3. Relaciones de la clase Product

Esta clase al ser una clase estructural, es utilizada también por las clases Mediator, Filters y los diferentes wrappers encargado de obtener los productos de los portales. Así sus relaciones son las siguientes:

- **Clase Wrapper:** Se relaciona con cada uno de los wrappers, que utilizarán su estructura para almacenar la información de los productos obtenidos en la búsqueda y enviarla a la capa Mediador.
- **Clase Filter**: esta clase crea listas de objetos de la clase Product, compara sus elementos, los ordena y los devuelve a la capa de Mediador.
- **Clase Mediator:** esta clase la utiliza para almacenar los productos que provienen de los wrappers y de la clase Filter y devolverlos a la clase View.

6.2.2.4. Clase SpecialSearchStrings

Esta clase es donde se implementa la funcionalidad de la creación de las cadenas de texto con las que se buscará en los portales seleccionados, en el caso de la búsqueda específica. En la Figura 24 podemos ver sus métodos.

Figura 24: Clase SpecialSearchStrings

Contiene métodos con la misma funcionalidad, pero específicos para cada portal, y para cada tipo de productos. Es decir hay tantos métodos como como productos y tiendas se han contemplado. Por esta razón solo se expondrán los ejemplos de los tipos de productos contemplados para un solo portal, Amazon.

6.2.2.4.1. Métodos de la clase SpecialSearchStrings

- **getStringForAmazonCameras:** procesa las opciones marcadas en caso de búsqueda personalizada de cámaras y crea la cadena de texto final que se utilizará para buscar en la página de Amazon.
- **getStringForAmazonCards:** procesa las opciones marcadas en caso de búsqueda personalizada de tarjetas de memoria y crea la cadena de texto final que se utilizará para buscar en la página de Amazon.
- **getStringForAmazonDesktop:** procesa las opciones marcadas en caso de búsqueda personalizada de ordenadores de sobremesa y crea la cadena de texto final que se utilizará para buscar en la página de Amazon.
- **getStringForAmazonLaptop:** procesa las opciones marcadas en caso de búsqueda personalizada de ordenadores portátiles y crea la cadena de texto final que se utilizará para buscar en la página de Amazon.
- **getStringForAmazonMP3:** procesa las opciones marcadas en caso de búsqueda personalizada de reproductores MP3 y crea la cadena de texto final que se utilizará para buscar en la página de Amazon.

- **getStringForAmazonPrinter:** procesa las opciones marcadas en caso de búsqueda personalizada de impresoras y crea la cadena de texto final que se utilizará para buscar en la página de Amazon.
- **getStringForAmazonScreen:** procesa las opciones marcadas en caso de búsqueda personalizada de monitores de ordenador y crea la cadena de texto final que se utilizará para buscar en la página de Amazon.

6.2.2.4.2. Relaciones de la clase SpecialSearchStrings

Clase Mediator: Esta clase se relaciona con la clase Mediator, recibiendo las peticiones de la misma para obtener las cadenas de texto creadas como resultado de las distintas características de los productos seleccionadas por el usuario en la búsqueda especifica. Una vez creadas, devuelve las cadenas a esta misma clase.

6.2.2.5. Clase SearchTimer

Esta clase es donde se implementa la funcionalidad de gestión del punto crítico de detectar una pérdida de conexión o una conexión lenta. Es una clase que extiende de la clase Threads de Java, e implementa el método en el que un proceso paralelo a la ejecución del programa se ejecuta cada vez que se inicia una búsqueda.

El proceso espera el tiempo indicado por el usuario en la interfaz para dar la conexión por perdida. Si la búsqueda termina antes del tiempo establecido, se termina el proceso, en otro caso, el proceso gestiona el funcionamiento de la aplicación, dando un mensaje de búsqueda errónea por problemas con la conexión. En la Figura 25 podemos observar sus métodos.

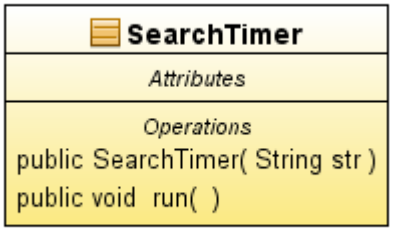

Figura 25: Clase SearchTimer

6.2.2.5.1. Métodos de la clase SearchTimer

- **SearchTimer():** crea un objeto de la clase Thread de Java, con el identificador pasado por parámetro.

- **run():** implementa la funcionalidad de esperar el tiempo indicando por el usuario y tras terminarse este tiempo dar por finalizada la búsqueda mostrando el mensaje que notifica de que existen problemas de la conexión.

6.2.2.5.2. Relaciones de la clase SearchTimer

Clase Mediator: se relaciona con Mediator, ya que esta clase es la que crea una instancia de la clase SearchTimer, tras recibir la petición de búsqueda. Esta instancia es eliminada cuando termina la operación de búsqueda.

6.2.3. Implementación del Nivel de Fuente de Datos

La implementación de esta capa es la más laboriosa, ya que hay que realizar un desarrollo distinto para cada una de los portales que hemos contemplado, esto es debido a la distinta estructura de código HTML que tiene cada uno de ellos.

Aunque los distintos wrappers tienen aparentemente los mismos métodos, luego en cada uno de ellos hay una implementación bastante distinta. Con lo cual se puede decir que la estructura de los wrappers es la misma pero no el contenido.

Así esta capa estará formada por las distintas clases que se corresponden con los diferentes wrappers asociados a los distintos portales y además un wrapper que obtendrá el tipo de conversión actual entre dólares y euros. En total son 8 clases (al haber contemplados 7 portales más el wrapper que obtiene el tipo de conversión). Como la estructura es similar en las 7 primeras, procederemos a explicar solo una de estas (la correspondiente al portal Amazon), además de explicar el wrapper dedicado a obtener el tipo de conversión, cuya funcionalidad difiere del resto.

6.2.3.1. Clase Amazonwrapper

En la Figura 26 podemos ver los métodos y atributos de la clase AmazonWrapper, la cual hemos elegido como representante para explicar estas características comunes a todas las clases que implementan los wrappers asociados a un portal.

Figura 26: Clase AmazonWrapper

Esta clase tiene varios atributos y un único método. Esto es así, porque la única función de los wrappers es la de realizar una búsqueda en el portal asociado. Con lo cual toda la lógica de esa búsqueda va en este método.

Procedemos pues, a explicar los atributos, métodos y relaciones de esta clase.

6.2.3.1.1. Atributos de la clase Amazonwrapper

- **url:** dirección web de la página principal de Amazon.
- **driver:** objeto de tipo WebDriver de las librerías de Selenium, utilizado para crear el objeto con el que poder navegar por el código HTML de la página de Amazón.
- **selenium:** objeto de las librerías de Selenium que nos ofrecerá los metodos necesarios para poder navegar por el código HTML de la página de Amazón.
- **productsName:** variable para guardar el nombre de los productos obtenidos en la búsqueda.
- **productsPrice:** variable para guardar el precio de los productos obtenidos en la búsqueda.
- **productsLink:** variable para guardar los links de las páginas de compra de los productos obtenidos en la búsqueda.
- **photoLinks:** variable para guardar los links de las fotos de los productos obtenidos en la búsqueda.

- **searchText:** cadena de texto a insertar en el buscador interno e Amazon.
- **secondCase:** variable auxiliar utilizada durante la obtención de información en la búsqueda para salvar diferentes peculiaridades ofrecidas por la estructura interna de la Web de Amazon.
- **numberResults:** número de productos obtenidos en la búsqueda.

6.2.3.1.2. Métodos de la clase Amazonwrapper

- **getInfo():** es el método que lleva toda la lógica de la búsqueda de los productos en la página de Amazon. Su función es la de realizar una búsqueda de los productos en el buscador general de Amazon, o en categorías especificas (en el caso de la búsqueda especifica) y extraer los resultados. Para ello utiliza la tecnología XPath y Selenium, explicadas en el capítulo 2 de este documento.

6.2.3.1.3. Relaciones de la clase Amazonwrapper

Los wrappers no se comunican entre sí. Solo se comunican con la clase Mediator para intercambio de información y también utilizan la clase Product para almacenar los productos obtenidos en la búsqueda. Así, sus relaciones son:

- **Clase Mediator:** obtienen de esta clase la información con la cadena de texto a buscar en el portal correspondiente. También envían los productos a esta clase una vez que han sido obtenidos en la búsqueda.
- **Clase Product**: Utilizan listas con elementos de la clase Product para almacenar los productos obtenidos, estas listas después serán enviadas a la clase Mediator.

6.2.3.2. Clase Exchangewrapper

Esta clase es la que implementa la funcionalidad de la obtención del tipo de conversión de moneda entre dólares y euros. Así la misión de este wrapper es navegar hacia la web de Google Finance, obtener el cambio actual entre estos dos tipos de monedas y almacenarlo para enviarlo al Nivel de Integración.

Procedemos pues, a explicar el único método y las relaciones de esta clase, que se pueden observar en la Figura 27.

Figura 27: Clase ExchangeWrapper

6.2.3.2.1. Métodos de la clase Exchangewrapper

- **getExchangeRate() :** el único método de la clase. Implementa la funcionalidad de la misma, es decir, utilizando las librerías de Selenium, navegar a través de la página de Google Finance, localizar el tipo de conversión actual entre dólares y euros, extraerlo y enviarlo al Nivel de Integración para su posterior procesamiento.

6.2.3.2.2. Relaciones de la clase Exchangewrapper

- **Clase Mediator**: La clase ExchangeWrapper recibe la petición de obtención del tipo de conversión entre euros y dólares de la clase Mediator y una vez obtenido le devuelve este valor a la misma.

6.2.3.3. Integración de wrappers en una única clase

Una de las mejores introducidas una vez realizada la aplicación, es la integración de las distintos wrappers en una sola clase. Así, las ocho clases contempladas para ellos han sido reducidas a una única clase en la que se almacena todo el código de navegación, búsqueda y extracción de resultados de cada uno de los portales. Con lo cual al realizar la petición de búsqueda, el mediador solo tendrá que ir llamando a diferentes métodos de una sola clase.

Esta mejora es sobre todo una mejora estructural, y se ha incluido para simplificar la estructura del diagrama de clases, aunque también se ha realizado para faci-

litar la inclusión de nuevos portales, ya que solo habría que meter el código de procesamiento de ese portal en esta clase en lugar de tener que crear una clase nueva.

En lugar de la integración se podría haber realizado la creación de una única interfaz para todos los wrappers. Aunque con esta opción tendríamos que mantener la clase ExchangeWrapper fuera de la implementación de esta interfaz ya que la estructura de la misma difiere respecto al resto de los wrappers, además de que todas las clases wrappers existentes se mantendrían.

Otra opción sería crear un único método en una clase wrapper única con un solo código de procesamiento común a todos los portales. Lo único que sería variante en este tipo de implementación serían las expresiones XPath referentes a los elementos de cada portal. Finalmente se ha desestimado esta opción por la complejidad que requería y la poca viabilidad de la misma, ya que no de todos los portales se extrae la misma información realizando las llamadas a los métodos que realizan esta extracción de la misma forma. Así dependiendo de la estructura de cada portal estas llamadas se realizarán de manera distinta, pudiendo ir o no dentro de distintos bucles, para salvar las peculiaridades de las estructuras de cada portal.

6.3. Características de la aplicación

6.3.1. Escalabilidad

Una característica que se ha pretendido cumplir en todo momento es que el sistema sea escalable. Esto significa que sea fácilmente ampliable a la hora de añadir nuevas características. En principio la escalabilidad se ha tenido en cuenta fundamentalmente para agregar nuevos portales de ventas de productos en un futuro.

Así, teniendo en cuenta la arquitectura utilizada, para agregar un nuevo portal solo hay que crear una entidad wrapper más en el Nivel de Fuente de Datos. Esta creación resultará muy sencilla dado que todos los wrappers cuentan con los mismos métodos. Lo único que cambiará será el procesamiento de extracción de datos, fundamentalmente el código XPath de acceso a los elementos del nuevo portal.

Como la estructura de almacenamiento de datos es homogénea, tampoco habrá cambios en este aspecto. Con lo cual teniendo en cuenta la implementación, solo habría que añadir una clase “NuevoPortalWrapper” al proyecto e implementar el único método que poseen los wrappers. Este procedimiento es bastante sencillo una vez conoces la estructura de los wrappers.

También comentar que los nombres de las clases, métodos y atributos se han escogido basándonos en el mantenimiento del proyecto. Así, todos ellos son muy

identificativos y cualquier persona ajena al proyecto tardará poco tiempo en entender que es lo que representa cada elemento.

6.3.2. Robustez

El sistema ha sido sometido a una serie de pruebas cuyo objetivo era dotar al sistema de la suficiente robustez como para poder considerarlo una aplicación suficientemente robusta.

Una salvedad en cuanto a los puntos críticos es la de los nexos de unión con los portales de venta de productos. Ya que como hemos explicado anteriormente para acceder a los distintos elementos HTML de estos portales necesitamos la ruta XPath de estos. Con lo cual, en el caso de que alguno de estos sitios cambie la estructura de su web y esto afecte a alguna ruta que utilizamos, la aplicación no podrá obtener los resultados de la búsqueda en ese portal.

Esta salvedad es un punto crítico externo que no podemos controlar y lo único que se puede realizar es solucionarlo lo antes posible una vez nos hayamos percatado del problema. Además, comentar que se ha intentado robustecer al máximo las expresiones XPath utilizadas para que se minimice este problema.

De cualquier manera, se ha comprobado que estos portales no son sometidos a apenas ningún cambio, ya que durante todo el proceso de desarrollo del proyecto, los cambios producidos en los mismos han sido mínimos (solo un wrapper en periodo de navidades cambio su estructura). Se presupone que el no producirse cambios en la estructura de estos portales viene motivado por:

- El buen funcionamiento de la estructura actual, por lo que no es necesario realizar mejoras de eficiencia.
- Estos portales gozan de una gran aceptación por lo que no es necesario arriesgarse a modificar su imagen.
- Los grandes cambios en estructuras que funcionan perfectamente, afectan a los usuarios que están acostumbrados ya a las mismas.

6.3.3. Reutilización

El objetivo de la reutilización es reducir costes en desarrollo y mantenimiento de un sistema. Uno de los aspectos principales de la reutilización es la evolución, el sistema debe ser flexible. El motivo por el que tiene que ser flexible es que si no lo fuera, si tuviera una estructura rígida, no se podría adaptar a un entorno cambiante,

por lo que al trasladarla a cualquier otro entorno que no sea el original, generaría muchos problemas

El concepto de reutilización se ha implementado en el proyecto desde un primer momento, ajustándonos a las reglas bien marcadas de la estructura elegida. De este modo, cualquier componente de la arquitectura independiente del nivel en el que se encuentre, se puede reutilizar de una manera muy efectiva y llevarlo a otro entorno de desarrollo para cualquier otro proyecto con una estructura similar. Por ejemplo el componente Mediador, podría ser utilizado para otro mashup de otras características distintas, ya que tiene todas las características y comunicaciones de un componente Mediador genérico.

6.3.4. Mantenimiento

La aplicación es un sistema que cuenta con un grado de autonomía muy alto, en principio no requiere de un gran mantenimiento, solo el destinado a cubrir los posibles errores motivados por los cambios en la estructura de la información de los portales contemplados.

Con el objetivo de que el mantenimiento del sistema sea lo más sencillo posible, se ha realizado la documentación de todas las clases y métodos utilizados en su implementación. Esto se ha realizado con ánimo de permitir agilizar los desarrollos necesarios para realizar futuras ampliaciones de la aplicación por parte de desarrolladores que lo desconozcan.

Así, la aplicación no requiere de mucho mantenimiento y puede ser perdurable en el tiempo, a no ser que los distintos portales de los que se sirve cambien su estructura y eso afecte a la aplicación. Aunque en estos casos, se ha preparado la aplicación para que solo haya que introducir las nuevas rutas Xpath de los elementos de los productos obtenidos en los resultados de la búsqueda, facilitando así la solución de este problema.

6.3.5. Portabilidad

En estos días en que existe una gran variedad de plataformas y sistemas operativos existentes la portabilidad es básica es una aplicación. Así el inconveniente de realizar una aplicación para una plataforma concreta radica en la imposibilidad de usuarios de otra plataforma de utilizarla, con lo cual nos estaríamos restando un gran número de potenciales usuarios.

Para prevenir esta situación, se ha implementado un sistema independiente de la plataforma. La portabilidad del sistema es garantizada gracias a que el sistema ha

sido implementado con lenguaje Java, lenguaje independiente de la plataforma, que permite realizar aplicaciones sin necesidad de tener en cuenta el sistema operativo sobre el cual se ejecutarán dichos programas.

El único requisito que se debe de tener para ejecutar la aplicación es tener instalada la máquina virtual de Java. Afortunadamente está máquina se puede instalar en cualquier sistema operativo.

Otros requerimientos necesarios para ejecutar la aplicación, que influyen en la portabilidad de la misma son:

- Conexión a Internet disponible para poder acceder a los diferentes sitios webs de donde se extraerá la información.
- Navegador Mozilla Firefox versión 3 o superior, que será lanzado por la aplicación con el objetivo de realizar las búsquedas. Actualmente existen versiones de Mozilla Firefox para los sistemas operativos más utilizados.

6.4. Uso de Selenium

En este apartado se van a describir con detalle todos los componentes de la tecnología Selenium (clases y métodos) utilizados en este proyecto, para que el lector se pueda hacer una idea del funcionamiento y comportamiento de cada uno de ellos, así como su función en la aplicación.

6.4.1. Clases utilizadas.

- **WebDriver:** esta clase representa el manejador del navegador web. Un objeto de esta clase define el navegador utilizado para realizar las interacciones. Posee métodos que realizan operaciones como lanzar el navegador, navegar hacia una URL concretar o realizar operaciones sobre elementos HTML.
- **FirefoxDriver:** es el manejador del navegador Mozilla Firefox. Permite hacer las operaciones que realiza un objeto WebDriver para este navegador determinado.
- **HtmlUnitDriver:** es la más rápida y ligera implementación de WebDriver. Se trata de un manejador para el navegador HtmlUnit, que es un navegador interno escrito en Java. Se ha utilizado en lugar de FirefoxDriver para el wrapper del portal Amazon.com, con el objetivo de aumentar la riqueza de tecnología en la implementación. No se ha utilizado en los demás wrappers porque presenta algunos problemas con el uso de JavaScript, con lo cual se prefirió utilizar FirefoxDriver en su lugar.

- **Selenium:** esta clase proporciona un extenso grupos de funcionalidades que van desde simular comportamientos del usuario con el navegador hasta la extracción de información de los elementos de la web actual.
- **WebDriverBackedSelenium:** es una clase que permite juntar el comportamiento de un objeto de tipo WebDriver junto con las funcionalidades de un objeto de tipo Selenium. Se utiliza para dotar a los manejadores de navegadores utilizados, las funcionalidades de la clase Selenium.

En la Figura 28 se observa cómo se han utilizado estas clases. Primero se crea el manejador del navegador Firefox, para ello tenemos que crear un objeto WebDriver que inicializamos al tipo FirefoxDriver, y finalmente, para juntar las funcionalidades del manejador con las ofrecidas por Selenium, creamos un objeto Selenium al que inicializamos como WebDriverBackedSelenium y le pasamos el manejador creado junto con la URL de inicio que queremos que se cargue cuando se lance el navegador.

```
static public WebDriver driver = new FirefoxDriver();

static public Selenium selenium = new WebDriverBackedSelenium(driver, "http://www.google.es/");
```

Figura 28: Clases Selenium utilizadas

6.4.2. Métodos utilizados

Los métodos utilizados son los proporcionados por el objeto “selenium” (cuya creación se puede ver en la Figura 28 que como hemos explicado proporciona los métodos de la clase Selenium a un objeto WebDriver. Estos métodos nos han servido para distintas funcionalidades como lanzar el navegador al iniciar la aplicación, navegar por los diferentes portales, realizar las búsquedas y obtener la información necesaria de los resultados de estas. Estos métodos son:

- **open(“url”):** lanza el navegador y abre la URL pasada por parámetro. Se utiliza en la carga de la aplicación para lanzar el navegador con el que se realizarán las búsquedas en los portales.
- **get(“url”):** método de la clase WebDriver que navega hacia la URL pasada por parámetro. Se ha utilizado para navegar por los distintos portales.
- **type(elemento, texto):** escribe el texto pasado como segundo argumento en el elemento HTML pasado como primero. Se utiliza para escribir el texto introducido por el usuario en las cajas de búsqueda de los distintos portales.

- **click(elemento):** pulsa (realiza un clic) el elemento HTML pasado por parámetro. Se utiliza para ejecutar el botón de búsqueda de cada portal, una vez introducido el texto en la caja.
- **getText(elemento):** obtiene el texto asociado al elemento HTML pasado por parámetro. Se utiliza para la obtención de los nombres y precios de los productos obtenidos al realizar las búsquedas.
- **getAttribute(elemento):** obtiene el valor del atributo asociado al elemento HTML pasado como parámetro. Se utiliza para obtener los enlaces de las páginas de los productos, así como los enlaces de sus fotografías.

En la Figura 29, se puede ver un ejemplo de la utilización de los métodos type() y click en el wrapper de Dynos.es, para escribir el texto introducido por el usuario en la caja de búsqueda y llevar a cabo esta operación.

```
ViewApp.selenium.type("//form[@id='f']/table/tbody/tr/td[1]/input", searchText);
ViewApp.selenium.click("//input[@value='Buscar']");
```

Figura 29: Ejemplo utilización métodos de Selenium

6.5. Gestión de errores

Se ha contemplado la gestión de errores de una manera sencilla, permitiéndonos saber en cada instante que error y donde se ha producido.

Al principio se pensó en realizar el control de errores almacenando los mismos en un fichero histórico, pero finalmente se descartó esta idea debido a los pocos puntos críticos que tiene la aplicación y a la carencia de utilidad para el usuario. Finalmente se ha implementado una gestión de errores en la cual, estos son notificados al usuario mediante un mensaje en la interfaz. En el apartado siguiente se explicarán los puntos críticos que tiene la aplicación desarrollada y como se han manejado cada uno de ellos. Finalmente, en el último apartado, se especificará como se ha implementado el manejo de excepciones.

6.5.1. Puntos críticos

Todos los posibles errores en estos puntos se han manejado para que el usuario sepa lo que está ocurriendo y pueda obrar en consecuencia, si así se requiere. Veamos en las siguientes tablas los puntos críticos de la aplicación en donde puede producirse un error durante su funcionamiento y como se han manejado.

Punto crítico	***Conexión lenta o perdida de la misma***
Descripción	El proceso de búsqueda puede quedarse en punto muerto o ralentizarse si la conexión a Internet de la máquina donde se está ejecutando la aplicación es muy lenta. Esto provoca que el funcionamiento de la aplicación pueda verse afectado hasta tal punto de parecer que no está siendo llevado a cabo.
Gestión	Este punto crítico se ha gestionado creando un proceso paralelo, mediante la utilización de hilos (clase Thread de Java), cada vez que se ejecuta una búsqueda, así cada proceso activará un contador que determina el tiempo límite que tiene para realizar la labor de navegación y extracción de información de los portales de la búsqueda. Si el tiempo excede este contador, se dará como una posible conexión lenta o pérdida de la misma y se mostrará un mensaje indicando el error de conexión al usuario, volviendo a la página principal del navegador y a la pestaña de búsqueda de la aplicación donde se encontraba anteriormente.

Tabla 37: Punto crítico: Conexión lenta o perdida

Punto crítico	***Introducción de un texto de búsqueda vacío***
Descripción	En la búsqueda genérica el usuario puede pulsar el botón de búsqueda sin haber escrito nada en la caja de texto correspondiente. Permitir esta acción provocaría realizar una búsqueda con texto vacío en los distintos portales.
Gestión	Si se introduce un texto de búsqueda vacío, no se permitirá su realización y se mostrara un mensaje indicando al usuario que debe de escribir el texto para poder realizar la búsqueda.

Tabla 38: Punto crítico: Introducción de un texto de búsqueda vacío

Punto crítico	***Productos o portales no seleccionados***
Descripción	En la búsqueda específica el usuario puede pulsar el botón de búsqueda sin haber seleccionado ningún producto de ninguno de los tipos ofrecidos. También podría ejecutar la operación y no haber seleccionado ningún portal donde realizarla (tanto en la búsqueda específica como en la genérica). Permitir estas acciones provocaría realizar una búsqueda errónea.
Gestión	Si no se selecciona alguna de estas opciones en la búsqueda, no se permitirá su realización y se mostrara un mensaje indicando al usuario que debe de seleccionar uno de los productos o en el caso de falta de portales, al menos un portal, para llevar a cabo la búsqueda.

Tabla 39: Punto crítico: Productos o portales no seleccionados

6.5.2. Manejo de excepciones

En este apartado explicaremos que se ha utilizado para manejar las posibles excepciones en el código de la aplicación. Una excepción se puede definir como un evento que ocurre durante la ejecución de la aplicación que interrumpe el flujo normal de las sentencias. Así, el manejo de excepciones lleva a cabo la gestión para condiciones anormales del flujo de ejecución de la aplicación, que pueden ser tratadas desde la misma.

En el lenguaje de programación utilizado, Java, el manejo de excepciones se realizar mediante objetos de clases derivadas de la clase Exception que a su vez deriva de la clase base Throwable. Así, esta clase tiene diferentes subclases para diferenciar el tipo de excepción que se ha producido, como por ejemplo la clase NullPointerException para indicar que se hace referencia a un objeto que no ha sido creado o ArrayIndexOutOfBound para indicar que se sobrepasa la dimensión de un array.

Estas excepciones se han controlado mediante el uso de las sentencias try-catch. Así diferenciamos dos bloques de código, el bloque de sentencias delimitado por “try” que ejecutará el código de la aplicación y el bloque de sentencias delimitado

por “catch” que se ejecutará en el caso de que en una sentencia contenida en el “try” se haya producido una excepción.

El objetivo de este control de excepciones es que cuando ocurra un comportamiento anómalo, el funcionamiento de la aplicación no termine y se ejecute un código adicional en su lugar.

Las excepciones producidas por el uso de la tecnología Selenium, se recogen en una clase específica llamada SeleniumException, que al igual que sus similares también deriva de la clase Exception de Java.

7. Evaluación de la aplicación

En esta sección se realizará una evaluación de la aplicación desarrollada en este proyecto. Se irán exponiendo los aspectos positivos y negativos de la aplicación.

Los aspectos positivos son aquellos en los que la aplicación destaca, sus puntos fuertes respecto a otras aplicaciones similares y respecto a la búsqueda manual de productos tecnológicos en los diferentes portales.

Para facilitar la explicación de los aspectos positivos y negativos, y exponerlos de una forma clara y concisa lo realizaremos mediante una tabla donde indicaremos el nombre de dicho aspecto y su descripción.

7.1. Aspectos positivos

Aspecto positivo	Descripción
Gran número y diversidad de fuentes	La aplicación consta de 7 portales como fuentes de búsqueda. Estas fuentes son muy representativas. Se han seleccionado las más importantes considerando los principales mercados internaciones y nacionales. La mayoría de las aplicaciones similares están centradas en 1 o 2 fuentes.
Rapidez	La tecnología Selenium, permite realizar las distintas búsquedas con una velocidad muy superior a la ofrecida por la búsqueda manual.
Independencia de la plataforma.	Al estar realizado en Java, la aplicación es ejecutable en cualquier sistema que tenga instalada la máquina virtual de Java. Con lo cual es independiente de la plataforma.
Búsqueda especializada	La búsqueda específica de productos distingue a nuestra aplicación de gran parte de sus semejantes, que por normal general no poseen esta característica.
Interfaz sencilla	La interfaz se caracteriza por su sencillez, lo que facilita mucho la interacción del usuario con la misma. Así es una interfaz muy simple, agradable para la vista, y no sobrecargada de elementos

	que podrían hacer más lenta su carga.
Conversión de moneda	La aplicación inicialmente está pensada para usuarios españoles, por eso se orientó el mercado a los mismos. Así para facilitar la compra a estos usuarios, se ha incluido la conversión de dólares a euros de las tiendas que no ofrecen este servicio. Otra característica que desmarca a la aplicación de gran parte de sus semejantes.
Orden	Otra de las facilidades incluidas para los usuarios de la aplicación, es el orden de los resultados según diversos criterios. Así los usuarios tienen la posibilidad de ordenar estos por relevancia, precio o portales.
Claridad en resultados	Los atributos de los productos elegidos para mostrar son los más representativos que un usuario puede desear ver. Así se obtiene una visión clara y concisa de los mismos, evitando atributos en un principio irrelevantes y que podrían llevar a confusión al usuario y a una sobrecarga de la interfaz.

Tabla 40: Aspectos positivos

7.2. Aspectos negativos

Aspecto negativo	Descripción
Dependencia de la estructura de las fuentes.	El procedimiento de extracción de información utilizado, depende altamente de la estructura de los portales que utilizamos como fuentes. Así, si los desarrolladores de estos portales cambian la ubicación de los elementos HTML contenidos en sus web, la aplicación no podrá extraer elementos de las mismas, no pudiendo realizar la búsqueda con éxito.
Dependencia de la conexión a Internet.	Evidentemente, para realizar las búsquedas se utiliza la conexión a Internet. Si la conexión es lenta o errónea en algún momento de la búsqueda, esto repercutirá en el funcionamiento de la aplicación, provocando una búsqueda más lenta, o en el

	caso de desconexión, una búsqueda errónea. Así el usuario puede elegir un tiempo máximo de búsqueda a partir del cual la aplicación daría la conexión por lenta o por perdida y se abortaría la operación.
Aplicación de escritorio	Actualmente la aplicación se encuentra implementada para ejecutarse como aplicación de escritorio. No obstante en las futuras mejoras se incluirá una versión que se pueda subir a un servidor y se ejecute directamente desde el navegador web.
Ejecución con un único navegador.	La aplicación realiza las búsquedas con el navegador Mozilla Firefox. Así, para el correcto funcionamiento de la misma, se necesita tener este navegador instalado. No obstante en las futuras mejoras se incluirá una versión para algunos de los navegadores más utilizados como Google Chrome o Microsoft Internet Explorer (actualmente los manejadores de Selenium para estos navegadores presentan problemas con Javascript).
Búsqueda no simultanea	La aplicación no realiza una búsqueda simultánea abriendo varias instancias del navegador, lo que aumentaría su rapidez. Al igual que con la mayoría de los inconvenientes subsanables, se incluirá como recomendación en las futuras mejoras, la implementación de la búsqueda simultánea en la aplicación.
Interfaz poco moderna.	Con el objetivo de simplificar la interfaz y de no sobrecarga mucho se eligió utilizar las clases SWING y AWT que ofrece Java para realizar interfaces. Así, la interfaz parece poco moderna y algo limitada en comparación con las interfaces de las aplicaciones actuales. Se incluirá como futura mejora la realización de una interfaz con más detalles cuidados y semejantes a los estilos actuales. Esta mejora podría realizarse para la versión Web de la aplicación.

Tabla 41: Aspectos negativos

8. Comparativa

En este apartado se va a realizar una comparativa de la aplicación de este proyecto con otras aplicaciones similares que ofrecen similar funcionalidad: la de realizar la búsqueda en diferentes sitios webs y reunir la información para mostrarla al usuario. Como también existe la opción de realizar esta funcionalidad manualmente con un navegador web (aunque sin agrupar todos los resultados), se ha realizado la comparativa con esta forma de búsqueda.

En cuantos a las aplicaciones escogidas, hay que decir que al haber un gran número de aplicaciones similares, han tenido que descartarse bastantes de ellas para realizar la comparativa. Las aplicaciones con las que se va a comparar han sido seleccionadas por su calidad y su utilización, además de por el grado de similitud con nuestra aplicación.

En cuantos a los parámetros y métricas utilizadas para comparar la aplicación, son varios y diversos. Se ha decidido mostrar estos parámetros para que el lector pueda tener información objetiva en la comparación. Además se incluirá un resumen con los comentarios acerca de los datos obtenidos en cada comparativa. En la siguiente tabla se puede ver la plantilla que se utilizará para mostrar estos parámetros. Resaltar que algunas opciones que no son aplicables para algunas comparativas irán señaladas con la notación N/A.

	Aplicación a comparar	**Aplicación del proyecto**
Tiempo en realizar una búsqueda en 1 fuente (segundos)	Nº de seg.	Nº de seg.
Tiempo en realizar una búsqueda en 2 fuentes (segundos)	Nº de seg.	Nº de seg.
Tiempo en realizar una búsqueda en 3 fuentes (segundos)	Nº de seg.	Nº de seg.
Tiempo en realizar una búsqueda en 4 fuentes (segundos)	Nº de seg.	Nº de seg.
Número de clics que tiene que realizar el usuario para una búsqueda de 1 fuente	Nº de clics	Nº de clics

Número de clics que tiene que realizar el usuario para una búsqueda de 2 fuentes	Nº de clics	Nº de clics
Número de clics que tiene que realizar el usuario para una búsqueda de 3 fuentes	Nº de clics	Nº de clics
Número de clics que tiene que realizar el usuario para una búsqueda de 4 fuentes	Nº de clics	Nº de clics
Número de fuentes contempladas	Nº Fuentes	Nº Fuentes
Plataformas disponibles	Nº Plataform.	Nº Plataform.
Soporta búsqueda especifica	Si/No	Si/No
Número de tipos de productos soportados en la búsqueda especifica	Nº de tipos.	Nº de tipos.
Número de atributos mostrados sobre los resultados	Nº Atributos	Nº Atributos
Número de criterios soportados para el orden de resultados	Nº Criterios	Nº Criterios
Conversión de precio de moneda extranjera a euros	Si/No	Si/No

Tabla 42: Plantilla de la comparativa

Comentar que las comparativas se han realizado utilizando el mismo navegador (tanto para la búsqueda manual como para las aplicaciones), en concreto Mozilla Firefox 3.6. También se ha utilizado el mismo ordenador y la misma conexión a Internet para reducir al mínimo el impacto que tengan estos factores a la hora de comparar los tiempos de búsqueda.

8.1. Comparativa con búsqueda manual

Esta comparativa ha sido realizada para obtener la ventaja que tiene el utilizar nuestra aplicación frente a realizar la búsqueda manualmente. Estas búsquedas se han realizado con el mismo procedimiento para cada fuente:

- Escribir la URL del portal en la barra de navegación.

- Seleccionar la caja de búsqueda del portal.
- Escribir el texto de búsqueda.
- Hacer clic en el botón de búsqueda.
- Esperar a obtener los resultados.
- Abrir otra pestaña y realizar los mismos pasos para la siguiente búsqueda.

Evidentemente, la muestra de resultados no es comparable, ya que este procedimiento no los agrupa ni los ordena en una sola interfaz. Por este motivo esta comparación se basa fundamentalmente en motivos de rendimiento (tiempo de búsqueda y número de clics realizados).

8.1.1. Tabla comparativa con la búsqueda manual

	Búsqueda manual	**Aplicación del proyecto**
Tiempo en realizar una búsqueda en 1 fuente (segundos)	8,3	4,5
Tiempo en realizar una búsqueda en 2 fuentes (segundos)	15,7	9,2
Tiempo en realizar una búsqueda en 3 fuentes (segundos)	23,4	14,4
Tiempo en realizar una búsqueda en 4 fuentes (segundos)	31,2	18,3
Número de clics que tiene que realizar el usuario para una búsqueda de 1 fuente	4	3
Número de clics que tiene que realizar el usuario para una búsqueda de 2 fuentes	8	4
Número de clics que tiene que realizar el usuario para una búsqueda de 3 fuentes	12	5
Número de clics que tiene que realizar el usuario para una búsqueda de 4 fuentes	16	6

Número de fuentes contempladas	N/A	7
Plataformas disponibles	Web	Java
Soporta búsqueda especifica	N/A	Si
Número de tipos de productos soportados en la búsqueda especifica	N/A	9
Número de atributos mostrados sobre los resultados.	N/A	4
Número de criterios soportados para el orden de resultados.	N/A	3
Conversión de precio de moneda extranjera a euros	No	Si

Tabla 43: Comparativa con la búsqueda manual

Como se puede observar en la tabla comparativa, el rendimiento de búsqueda de nuestra aplicación es muy superior a la búsqueda manual tanto en el aspecto temporal como en el número de clics pulsados por el usuario. Esta superioridad se acrecienta según va aumentando el número de fuentes.

Tanto estos resultados de rendimiento como las numerosas ventajas que ofrece nuestra aplicación frente a la búsqueda manual (como la agrupación de resultados de distintas fuentes en un solo formato o la conversión de moneda) constatan de forma evidente los beneficios de utilizar nuestra aplicación frente a este tipo de búsqueda.

8.2. Comparativa con aplicaciones similares

La comparativa se ha realizado con 3 aplicaciones similares a la presentada en este proyecto, estas son Q-compare.com, FlatTVPrice.com y GotFreeShipping.com, todas ellas en versión Web. Pasamos a explicar cada una de estas comparativas.

8.2.1. Comparativa con Q-compare.com

Esta aplicación es un mashup comercial cuya función actual es ser un comparador de precios, entre Amazon y Ebay. Realiza búsquedas en Amazon primero, y luego tras seleccionar un producto obtenido como resultado, se realiza comparación del mismo con su oferta en Ebay. Esto es útil para ver si hay algún artículo en Ebay idéntico, de segunda mano, a un mejor precio que el artículo de Amazon. La web donde se aloja se puede observar en las Figuras 30, 31 y 32.

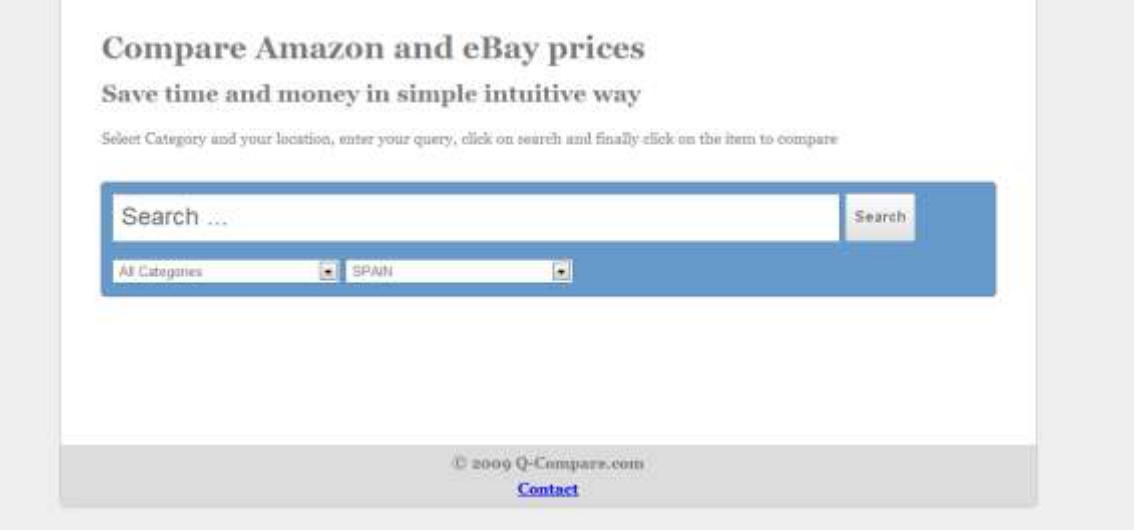

Figura 30: Página principal de QCompare.com

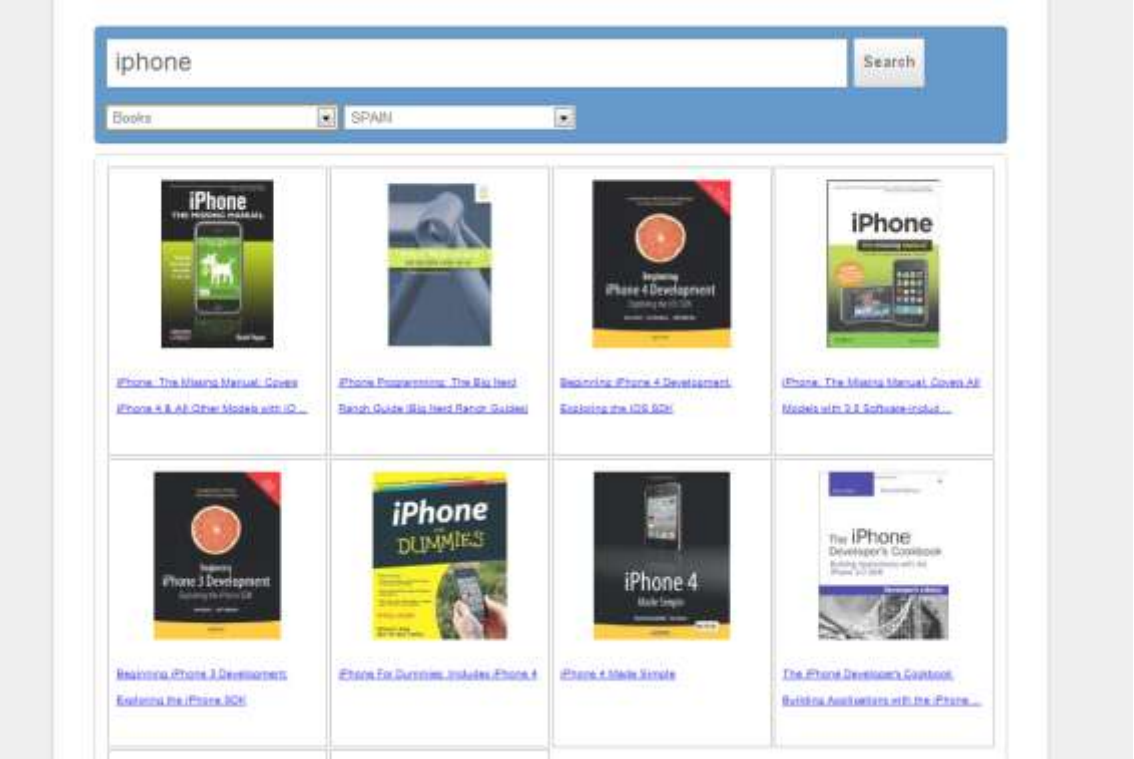

Figura 31: Página de resultados de Q-Compare.com

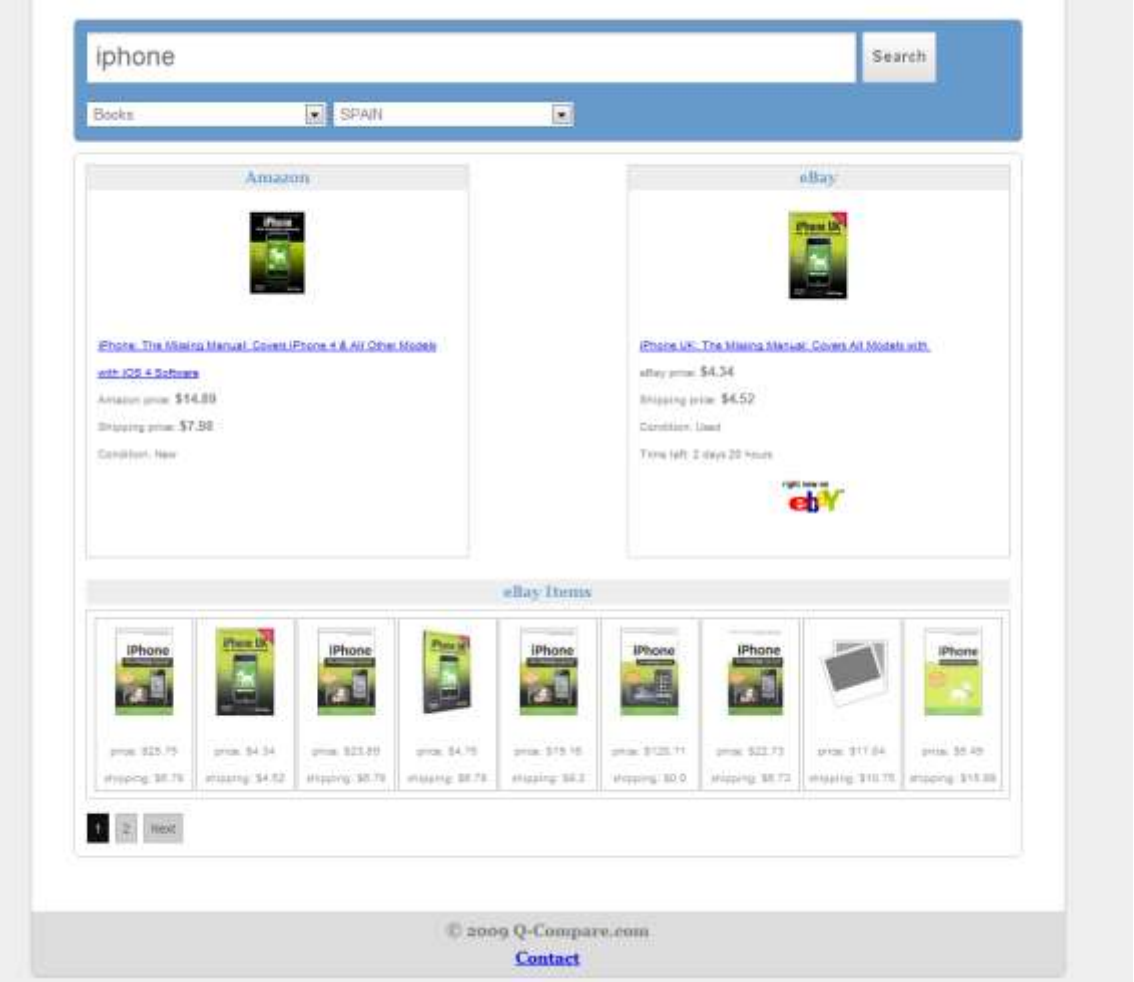

Figura 32: Página de comparativa de Q-Compare.com

Las características de la aplicación son las siguientes:

- Soporta búsqueda por categorías, aunque no funciona de manera muy eficiente, ya que muestra productos en categorías que no son la suya.
- Muestra la foto de los productos en un tamaño bastante bueno.
- Incluye gastos de envió dependiendo del país seleccionado.
- Muestra un número de productos muy elevado.
- Tiene un comparador del mismo producto en Ebay como en Amazon. Por lo que debe de asegurarse que el producto sea el mismo. Detecta productos únicos.
- La interfaz gráfica pese a estar realizada en web, es muy básica.

Antes de ver la comparativa comentar que esta aplicación nunca realiza dos búsquedas a la vez, ya que primero busca en Amazon y te devuelve sus productos y luego al seleccionar uno de ellos, busca los productos similares en Ebay, con lo cual el procedimiento de búsqueda no es continuo como si lo es en nuestra aplicación.

Pasemos a ver la comparativa:

8.2.2. Tabla comparativa con Q-Compare.com

	Q-Compare	**Aplicación del Proyecto**
Tiempo en realizar una búsqueda en 1 fuente (segundos)	3,3	4,5
Tiempo en realizar una búsqueda en 2 fuentes (segundos)	6,8	9,2
Tiempo en realizar una búsqueda en 3 fuentes (segundos)	N/A	14,4
Tiempo en realizar una búsqueda en 4 fuentes (segundos)	N/A	18,3
Número de clics que tiene que realizar el usuario para una búsqueda de 1 fuente	2	3
Número de clics que tiene que realizar el usuario para una búsqueda de 2 fuentes	4	4
Número de clics que tiene que realizar el usuario para una búsqueda de 3 fuentes	N/A	5
Número de clics que tiene que realizar el usuario para una búsqueda de 4 fuentes	N/A	6
Número de fuentes contempladas	2	7
Plataformas disponibles	Web	Java
Soporta búsqueda especifica	Si	Si

Número de tipos de productos soportados en la búsqueda especifica	19	9
Número de atributos mostrados sobre los resultados.	1	4
Número de criterios soportados para el orden de resultados.	0	3
Conversión de precio de moneda extranjera a euros	No	Si

Tabla 44: Comparativa con Q-Compare.com

De esta comparativa observamos que el punto fuerte en el que Q-Compare supera a nuestra aplicación es la velocidad. También hay otros puntos positivos de esta aplicación como el incluir los gastos de envío para el país seleccionado o la posibilidad de búsqueda específica según 19 categorías, aunque no incluye características de estos productos y además son las categorías de Amazon por defecto, cosa muy factible al ser un mashup que realiza la primera búsqueda solo en ese portal. También resaltar su ejecución en web, que le hace accesible desde cualquier navegador.

Sin embargo nuestra aplicación, pese a ser algo más lenta, soporta búsquedas simultaneas de hasta 7 fuentes, muestra el precio de los productos en la primera pantalla de resultados, mientras que en Q-Compare tienes que hacer clic sobre el producto específico (solo muestra el nombre en la pantalla de resultados). Además incluye conversión de moneda y criterios de ordenación de resultados, opciones que no soporta Q-Compare.

Como conclusión decir que si se quiere comprar un producto determinado y solo comparar en las dos fuentes soportadas por Q-Compare: Amazon y Ebay, es mejor utilizar Q-Compare. Pero para comparar el mismo producto o similares de uno encontrado en Amazon con otros portales, tiene muchas más ventajas utilizar nuestra aplicación, por todos los motivos explicados anteriormente.

8.2.3. Comparativa con Flattvprice.com

Esta aplicación Web es un mashup que busca diferentes modelos de televisiones, y al seleccionar uno específico, te busca su precio en distintos portales de venta de estos artículos.

Al estar especializado en un tipo de artículo, recoge muy bien las características de los productos, y puede realizar una búsqueda más precisa. También recoge las principales marcas en su interfaz principal para realizar una búsqueda de artículos exclusivos de esas marcas.

El procedimiento de búsqueda es el siguiente: el usuario realiza una búsqueda de un artículo, la aplicación devuelve los resultados de la misma y el usuario deberá seleccionar uno de ellos. Para ese resultado se compararán los precios en diferentes portales de venta del artículo.

El aspecto de esta aplicación viene descrito en las Figuras 33 y 34.

Figura 33: Página principal de Flattvprice.com

Figura 34: Página de resultados de Flattvprice.com

Las características principales de esta aplicación son:

- Posibilidad de seleccionar la marca y el modelo sin tener que introducirlo textualmente.
- Seguridad de comparar precios del mismo producto y no de productos similares.
- Incluye fotografía en un tamaño bastante aceptable.
- Incluye búsquedas recientes.
- Recoge las valoraciones y opiniones de los productos.

Comentar, que no son conocidas el número de fuentes que utiliza, con lo cual la comparativa en cuanto al número de fuentes no es posible. Pero lo hemos simulado realizando una búsqueda de un producto para el que FlatTVPrice lo encuentra en 3 portales distintos (no sabemos si buscará en más de antemano). Así igualaremos los 3 portales en nuestra aplicación y descartaremos realizar búsquedas con otro número diferentes de portales. Pasemos a ver la tabla comparativa.

8.2.3.1. Tabla comparativa con FlatTVPrice.com

	FlatTVPrice	Aplicación del Proyecto
Tiempo en realizar una búsqueda en 1 fuente (segundos)	N/A	4,5
Tiempo en realizar una búsqueda en 2 fuentes (segundos)	N/A	9,2
Tiempo en realizar una búsqueda en 3 fuentes (segundos)	8	14,4
Tiempo en realizar una búsqueda en 4 fuentes (segundos)	N/A	18,3
Número de clics que tiene que realizar el usuario para una búsqueda de 1 fuente	3	3
Número de clics que tiene que realizar el usuario para una búsqueda de 2 fuentes	3	4
Número de clics que tiene que realizar el usuario para una búsqueda de 3 fuentes	3	5
Número de clics que tiene que realizar el usuario para una búsqueda de 4 fuentes	3	6
Número de fuentes contempladas	N/A	7
Plataformas disponibles	Web	Java
Soporta búsqueda especifica	Si	Si
Número de tipos de productos soportados en la búsqueda especifica	20 (marcas de televisiones)	9
Número de atributos mostrados sobre los resultados.	7	4

Número de criterios soportados para el orden de resultados.	0	3
Conversión de precio de moneda extranjera a euros	No	Si

Tabla 45: Comparativa con FlatTVPrice.com

Como conclusión de esta comparativa, observamos que al igual que Q-Compare, el punto fuerte en el que FlatTVPrice supera a nuestra aplicación es la velocidad. Otros puntos fuertes son la seguridad de estar comparando el mismo producto, esto resulta más sencillo de implementar, debido a la especificidad de este mashup (solo se centra en televisiones planas). Otras ventajas adicionales son la inclusión de opiniones y valoraciones de los productos (muy apreciado por los usuarios), así como su ejecución en web, que le hace accesible desde cualquier navegador.

Por el contrario nuestra aplicación superar a este mashup en otros aspectos, el principal y más evidente es la amplia variedad de productos ofrecidos. En cuanto al número de fuentes soportadas, aunque no sabemos con certeza cuantas soporta FlatTVPrice, todo parece indicar (tras realizar múltiples búsquedas) que no tienen que ser más de 7, que precisamente son las búsquedas simultaneas que soporta nuestra aplicación.

Otras funcionalidades que ofrece nuestra aplicación con respecto a FlatTVPrice, son la posibilidad de ordenar los resultados y la muestra de una gran cantidad de productos en la primera pantalla que aparece tras la búsqueda. Esta muestra se realiza con todos los atributos de los productos, mientras que en FlatTVPrice tienes que hacer clic sobre el producto específico. También comentar que FlatTVPrice no incluye conversión de moneda.

Además comentar que FlatTVPrice compara un producto único, lo que a veces puede ser una ventaja o una desventaja ya que no permite ver en la misma comparación productos similares (acción que si permite nuestra aplicación).

Como conclusión, FlatTVPrice es mejor opción si lo que queremos es comprar un producto (una televisión plana) especifico. Así, ya deberíamos tener claro el producto a comprar, y utilizar simplemente FlatTVPrice como comparador de precio. Pero cuando no tenemos claro el producto o no es el tipo producto específico que soporta FlatTVPrice (las televisiones planas), el uso de nuestra aplicación presenta más ventajas.

8.2.4. Comparativa con Gotfreeshipping.com

Esta aplicación busca todo tipo de productos por categorías en Ebay que tengan los gastos de envió gratis. Pese a ser una aplicación especializada en una sola fuente, hay que decir que aprovecha esta muy bien por varios motivos:

- Puede ofrecer más información común a todos los productos (como leer los gastos de envío y seleccionar los que sean coste cero).
- Soporta multitud de categorías (todas las soportadas en Ebay).
- También muestra información sobre las pujas realizadas en Ebay sobre los productos.

El diseño es muy original y soporta múltiples idiomas, lo que amplía el número de potenciales usuarios. También soporta el orden de los resultados por diversos criterios como:

- Relevancia
- Productos más recientes.
- Productos más antiguos.
- Productos con más pujas realizadas.

El procedimiento de búsqueda es el mismo que la mayoría de los mashups, hay una caja de texto en la que se introduce el texto a buscar y realizando clic en el botón de búsqueda obtenemos los productos como resultado. También soporta búsqueda por categorías, en la que solo tenemos que seleccionar la categoría en lugar de escribir el texto de búsqueda para obtener los resultados. De estos productos solo se muestran la foto y el precio, esto bajo mi punto de vista es original pero poco práctico, ya que en los resultados es difícil saber cuál es el modelo de cada producto. Su interfaz se muestra en las Figuras 35 y 36.

Figura 35: Página principal de Gotfreeshipping.com

Figura 36: Página de resultados de Gotfreeshipping.com

Resumiendo, las características más importantes de este mashup son:

- Interfaz original
- Presentación de productos sin nombre (solo foto y precio).

- Multitud de categorías.
- Búsquedas relacionadas.
- Soporte de diversos idiomas.
- Específico para una fuente (Ebay).

Teniendo en cuenta las diferencias que presenta con nuestra aplicación, presentamos la tabla comparativa.

8.2.4.1. Tabla comparativa con Gotfreeshipping.com

	Gotfreeshipping.com	**Aplicación del proyecto**
Tiempo en realizar una búsqueda en 1 fuente (segundos)	2,3	4,5
Tiempo en realizar una búsqueda en 2 fuentes (segundos)	N/A	9,2
Tiempo en realizar una búsqueda en 3 fuentes (segundos)	N/A	14,4
Tiempo en realizar una búsqueda en 4 fuentes (segundos)	N/A	18,3
Número de clics que tiene que dar el usuario para una búsqueda de 1 fuente	2	3
Número de clics que tiene que dar el usuario para una búsqueda de 2 fuentes	N/A	4
Número de clics que tiene que dar el usuario para una búsqueda de 3 fuentes	N/A	5
Número de clics que tiene que dar el usuario para una búsqueda de 4 fuentes	N/A	6
Número de fuentes contempladas	N/A	7

Plataformas disponibles	Web	Java
Soporta búsqueda especifica	Si	Si
Número de tipos de productos soportados en la búsqueda especifica	Más de 100.	9
Número de atributos mostrados sobre los resultados.	2	4
Número de criterios soportados para el orden de resultados.	4	3
Conversión de precio de moneda extranjera a euros	No	Si

Tabla 46: Comparativa con Gotfreeshipping.com

De esta tercera comparación extraemos una conclusión similar a las dos anteriores. Así observamos que el punto fuerte en el que Gotfreeshipping.com supera a nuestra aplicación es la búsqueda específica por categorías, ya que soporte un gran número de ellas. Además al igual que las demás aplicaciones, la velocidad de búsqueda es superior a la de nuestra aplicación.

Un aspecto muy importante de este mashup es la originalidad de la interfaz. Como se puede observar la pantalla principal está llena de etiquetas con el nombre de las categorías, lo que haciendo clic en ellas permite hacer una búsqueda en cada una. También los resultados son mostrados de forma original, dando mucha importancia a las fotos. Esto es un arma de doble filo, pues no incluyen el nombre ni descripción del producto, con lo cual el usuario debe conocer el artículo por la foto.

Al igual que el resto de las aplicaciones comparadas, su ejecución es en web, que le hace accesible desde cualquier navegador, punto muy importante. También resaltar que los productos se pueden ordenar por un criterio más que nuestra aplicación.

Los puntos en los que nuestra aplicación ofrece mayor funcionalidad son la gran variedad de fuentes y la muestra de diversos atributos de los resultados. Esta última hace tener al usuario una idea más clara de que productos concretos se están mostrando en la interfaz.

La conclusión obtenida es la misma que en las aplicaciones comparadas anteriormente. Si se quiere comprar un producto en Ebay con gastos de envío gratis, está bien utilizar Gotfreeshipping.com, es decir en su funcionalidad específica, esta aplicación mejora sustancialmente a la nuestra. Pero el no soportar distintas fuentes, limita mucho las funcionalidades de la misma.

8.3. Tabla comparativa genérica

A continuación resumimos la comparativa con los tres mashups elegidos y la búsqueda manual en la siguiente tabla. Así se puede obtener una vista rápida de las principales diferencias entre estas aplicaciones y la implementada en este proyecto.

	Búsqueda Manual	**FlatTVPrice**	**Q-Compare**	**Gotfreeshipping.com**	**Aplicación del proyecto**
Tiempo en realizar una búsqueda en 1 fuente (segundos)	8,3	N/A	3,3	2,3	4,5
Tiempo en realizar una búsqueda en 2 fuentes (segundos)	15,7	N/A	6,8	N/A	9,2
Tiempo en realizar una búsqueda en 3 fuentes (segundos)	23,4	8	N/A	N/A	14,4
Tiempo en realizar una búsqueda en 4 fuentes (segundos)	31,2	N/A	N/A	N/A	18,3
Número de clics que tiene que realizar el usuario para una búsqueda de 1 fuente	4	3	2	2	3
Número de clics que tiene que realizar el usuario para una búsqueda de 2 fuentes	8	3	4	N/A	4

Número de clics que tiene que realizar el usuario para una búsqueda de 3 fuentes	12	3	N/A	N/A	5
Número de clics que tiene que realizar el usuario para una búsqueda de 4 fuentes	16	3	N/A	N/A	6
Número de fuentes contempladas	2	N/A	2	N/A	7
Plataformas disponibles	Web	Web	Web	Web	Java
Soporta búsqueda especifica	N/A	Si	Si	Si	Si
Número de tipos de productos soportados en la búsqueda especifica	N/A	20 (marcas de televisiones)	19	Más de 100.	9
Número de atributos mostrados sobre los resultados.	N/A	7	1	2	4
Número de criterios soportados para el orden de resultados.	N/A	0	0	4	3
Conversión de precio de moneda extranjera a euros	N/A	No	No	No	Si

Tabla 47: Tabla comparativa general

9. Conclusiones, valoraciones y futuras mejoras

9.1. Objetivos cumplidos

El objetivo principal de este proyecto era el desarrollo de una herramienta que pudiera realizar búsquedas en diferentes portales Web de venta de productos tecnológicos, facilitando a los usuarios la tarea de búsqueda y comparación de estos productos.

Una vez finalizado el proyecto es necesario echar la vista atrás y recordar los objetivos planteados al principio de este, para observar hasta qué punto se han cumplido las expectativas. En la siguiente tabla, se expondrán estos objetivos generales y será comentada la forma en las que estos se han cumplido.

Objetivo inicial
Integrar bajo una misma aplicación el contenido de los productos tecnológicos de los portales de los sectores más importantes y más frecuentemente utilizados tanto a nivel mundial como nacional.
Realizado
La integración se ha realizado aceptablemente, ya que se han considerado los portales de este sector más importantes del momento, tanto a nivel internacional como nacional. Así la aplicación finalmente realiza esta búsqueda en estos portales e integra en ella todo el contenido obtenido de los mismos.

Tabla 48: Objetivo inicial 1

Objetivo inicial
Mostrar la información más relevante de cada producto.
Realizado
Tras descartar algunos datos irrelevantes y recolectar la información común a todos los portales. La información de cada producto mostrada finalmente es: portal que lo oferta, fotografía, nombre y precio. Finalmente no se ha incluido demasiada información complementaria debido a la sobrecarga que la interfaz podría sufrir, pero la información básica está completa.

Tabla 49: Objetivo inicial 2

Objetivo inicial
Estudiar la posibilidad de implementar una aplicación capaz de interactuar con el navegador Web de manera inteligente.
Realizado
La navegación inteligente se ha realizado en la aplicación mediante el uso de la tecnología Selenium. Así la aplicación controla en todo momento esta interacción con el navegador enviándole las instrucciones necesarias para navegar entre los distintos portales, seleccionar sus elementos e interactuar con ellos, como si de un usuario se tratara.

Tabla 50: Objetivo inicial 3

Objetivo inicial
El estudio de la tecnología Selenium para llevar a cabo la navegación automática.
Realizado
Como se ha comentado en el objetivo anterior, la tecnología empleada para la navegación automática fue Selenium. Así, se tuvo que realizar un estudio de esta tecnología, con sus distintas posibilidades, compatibilidades y funciones. El estudio resulto positivo al haber servido para poder utilizar esta tecnología para nuestro propósito.

Tabla 51: Objetivo inicial 4

Objetivo inicial
Que la interfaz de la aplicación cumpliera con los propósitos de accesibilidad y navegabilidad, para facilitar así el acceso al a misma a los distintos tipos de usuario que puedan beneficiarse de ella.
Realizado
Este aspecto ha sido cubierto aceptablemente. La interfaz es muy simple, no cargada de elementos, y optimizada para pantallas iguales o superiores a 15 pulgadas, lo que supone gran parte de los ordenadores del mercado actuales.

Tabla 52: Objetivo inicial 5

9.2. Valoración económica

A continuación vamos a mostrar el presupuesto del proyecto. Primero mostraremos un resumen con los datos numéricos más importantes del coste total de la ejecución del proyecto (tanto datos temporales como económicos). Después detallaremos mediante una tabla este presupuesto, considerando el tiempo empleado y material necesario para la elaboración del proyecto.

9.2.1. Datos temporales generales

En la siguiente tabla se muestran los datos temporales generales del desarrollo del proyecto:

Fecha de inicio del proyecto	2 de Agosto de 2010
Fecha de fin del proyecto	30 de Abril de 2011
Duración del proyecto en días empleados (sin incluir fines de semana y festivos)	187 días
Horas por día dedicadas	4 horas
Duración del proyecto en horas empleadas	748 horas

Tabla 53: Datos temporales generales

9.2.2. Datos económicos generales

En la siguiente tabla se muestran los datos económicos generales de la ejecución del proyecto:

SUBTOTAL MATERIAL	1.640,00€
SUBTOTAL INGENIERÍA Y DESAROLLO	23.740,00€
TOTAL	25.380,00€

Tabla 54: Datos económicos generales

9.2.3. Presupuesto en material

En la siguiente tabla desglosaremos el presupuesto de la ejecución del proyecto, en cuanto al material:

Concepto	Referencia	Coste	Uds	Total
Equipo informático	Sony Vaio S VGN-N38Z	1.200,00€	1	1.200,00€
Monitor 21''	HP	123,00€	1	123,00€
Licencia Windows 7	Windows 7 Professional	317,00€	1	317,00€
Licencia Netbeans		0,00€	1	0,00€
Licencia Selenium		0,00€	1	0,00€
		SUBTOTAL		1.640,00€

Tabla 55: Presupuesto en material

9.2.4. Presupuesto de ingeniería y desarrollo

En la siguiente tabla desglosaremos el presupuesto de la ejecución del proyecto, en cuanto a la ingeniería y desarrollo:

Concepto	Precio/hora	Nº horas	Uds	Total
Definición y organización del proyecto	50€	54	1	2.720,00€
Generación de especificaciones	40€	20	1	800,00€
Generación de documentación	30€	210	1	6.300,00€
Desarrollo del sistema	30€	464	1	13.920,00€
		SUBTOTAL		23.740,00€

Tabla 56: Presupuesto en ingeniería y desarrollo

9.3. Futuras mejoras de la aplicación

Al finalizar la implementación del proyecto y al hacer una evaluación del mismo, se han recogido las posibles futuras mejoras que pueden llevar a un desarrollo más sofisticado de la misma. Así, estas mejoras sobretodo están basadas en subsanar los puntos débiles que presenta la aplicación y realizar así una versión mejorada.

Se ha evaluado la factibilidad de las propuestas aquí realizadas, con lo cual, su implementación es posible aunque en algunos casos sea costosa de llevar a cabo.

Hubiera sido muy interesante realizar para este proyecto estas mejoras, pero la complejidad del mismo se elevaría bastante, ya que la mayoría de ellas requieren de bastante tiempo de implementación. Por esta razón se ha decido incluirlas en esta lista, que sin más dilación se procede a presentar.

9.3.1. Aplicación Web

La implementación de una aplicación con tecnología web sería una mejora bastante importante, que daría un salto de calidad a la aplicación actual (actualmente solo está implementada la versión de escritorio). Así, se implementaría la opción de subir la aplicación a un servidor y poder ejecutarla desde cualquier ordenador sin necesidad de instalarla. Para realizar esta mejora se puede optar por múltiples variantes, enumeraremos las dos principales que se han contemplado:

- **HTML + JavaScript:** realización de una página web con código JavaScript desde donde se llamarían a las clases Java de nuestra aplicación.
- **Applet Java:** los Applet Java son aplicaciones realizadas en Java que se pueden ejecutar en un navegador utilizando la Java Virtual Machine . Así reutilizando en su entereza el código de la aplicación, la aplicación podría ser ejecutada desde un navegador.

9.3.2. Soporte para múltiples navegadores

Actualmente solo se da soporte a uno de los navegadores más utilizados: Mozilla Firefox. Pero también, la tecnología Selenium permite realizar sus funcionalidades para otros navegadores como Google Chrome o Microsoft Internet Explorer.

Este soporte fue intentado implementar desde el primer momento, debido a su sencillez, pero el problema es que los manejadores de Selenium para estos navegadores, no funcionaban bien del todo con JavaScript, lo que supone que las búsque-

das no se hagan correctamente en algunos portales. Así que se obtuvo demasiados problemas y por esta razón finalmente se descartó su inclusión. Selenium está mejorando este soporte constantemente y es factible que saquen una versión mejorada de estos controladores.

9.3.3. Búsqueda simultánea

La posibilidad de búsqueda simultanea es una posibilidad real utilizando la tecnología Selenium, aun así, se descartó por la relación entre la gran complejidad que suponía implementarla y el no tan elevado rendimiento que iba a ganar la aplicación. Finalmente sea una búsqueda simultánea o no, la rapidez depende de la conexión a Internet, con lo cual si se lanzaran 7 búsquedas a la vez, en lugar de una, la velocidad de la búsqueda no sería 7 veces superior. El parámetro del que depende la rapidez es de la velocidad de la conexión. Aun así, se obtendría una pequeña mejora en su rendimiento, por esta razón su implementación sería interesante de realizar.

9.3.4. Interfaz actual

La interfaz realizada se presenta muy limitada y un poco anticuada, debido a las limitaciones de la tecnología empleada para su realización. Aunque esto favorece su sencillez y eficiencia, se considera evidente que una mejora gráfica sería muy bien valorada y acogida por parte del usuario.

Así la realización de esta interfaz se podría realizar en la versión de la aplicación Web, ya que supondría realizar una interfaz completamente nueva. La nueva interfaz debería estar en concordancia con las interfaces ofrecidas por las aplicaciones actuales.

9.3.5. Aplicación para dispositivos móviles

Actualmente Selenium está trabajando para realizar controladores para dispositivos móviles como teléfonos iPhone o aquellos que incorporan el sistema operativo Android. La realización de una versión de la aplicación para estos terminales sería dar un gran salto en cuanto al número de potenciales usuarios que puedan utilizar la aplicación, ya que cada vez este tipo de dispositivos es más utilizado.

Esta mejora todavía no es factible ya que no existe la tecnología actual para ello, pero se prevé que exista en un futuro cercano. La mayor complejidad de realizar

esta propuesta es el rediseño de toda la interfaz para pantallas de teléfonos móviles. Así habría que adaptar la muestra de resultados, como también las demás ventanas de la aplicación, para que se visualice fácilmente en estos dispositivos. Vamos a analizar cómo se podría realizar la implementación de la interfaz para la aplicación si se hubiera desarrollado su versión para dispositivos móviles.

9.3.5.1. Implementación de la interfaz para dispositivos móviles

Analizaremos dos tecnologías para implementar esta interfaz: J2ME y Android, en las cuales se puede utilizar la mayoría del código escrito en la aplicación de escritorio desarrollada en Java. Solo habría que crear una interfaz nueva con ellas e integrarla con el código de nuestra aplicación (además de utilizar tecnología Selenium para dispositivos móviles).

9.3.5.1.1. Interfaz en J2ME

J2ME (Java 2 Micro Edition) es la tecnología basada en la plataforma Java que ofrece una colección de APIs para el desarrollo de aplicaciones para dispositivos móviles como por ejemplo teléfonos, MP4s, PDAs…

Las librerías utilizadas por Java para hacer la interfaz (Swing y AWT) son muy pesadas para utilizarlas en dispositivos móviles, por ello J2ME trae otra librería que nos ofrece unos componentes específicos y más ligeros para teléfonos móviles. Este API estándar está el contenido en el paquete: “javax.microedition.lcdui”. Las posibilidades que nos ofrece son muy limitadas, pues sus componentes son muy estáticos y son un poco pobres gráficamente (En la Figura 37 podemos ver una interfaz realizada con componentes ofrecidos por esta API).

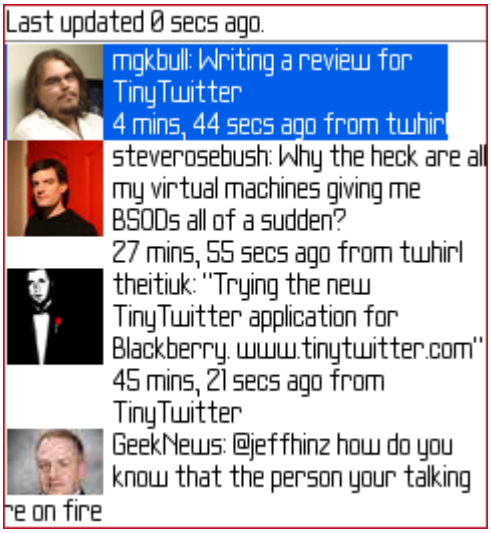

Figura 37: Interfaz realizada en J2ME

Aunque esta API no hay que descartarla, sí que conviene evaluar otras alternativas para poder realizar una aplicación más acorde con las actuales, utilizando J2ME. Así, existen una serie de alternativas a esta API por defecto, que ofrecen más posibilidades visuales a la hora de implementar la interfaz gráfica con esta tecnología. Vamos a proceder a describirlas continuación:

- **KUIX:** Es un framework de desarrollo para la creación de aplicaciones en J2ME. Este framework ofrece componentes gráficos más avanzados de los elementos más utilizados en la creación de interfaces como botones, campos de texto, listas... Un ejemplo de interfaz gráfica hecha con KUIX es el de la siguiente figura:

Figura 38: Interfaz realizada en J2ME utilizando KUIX

- **LWUIT:** Es una API que al igual que KUIX ofrece unos componentes gráficos avanzados y visualmente atractivos. Además soporta algunos efectos visuales como animaciones y transiciones. Un ejemplo de interfaz gráfica hecha con LWUIT es el de la Figura 39:

Figura 39: Interfaz realizada en J2ME utilizando LWUIT

- **J2ME Polish:** Es una suite de herramientas que incluye librerías para desarrollar aplicaciones completas. Una de estas librerías es una potente API para crear interfaces gráficas. Esta librería se llama LUSH y ofrece una gran cantidad de componentes muy atractivos visualmente. También soporta efectos gráficos como transiciones y animaciones. Un ejemplo de interfaz gráfica hecha con J2ME Polish es el de la siguiente figura:

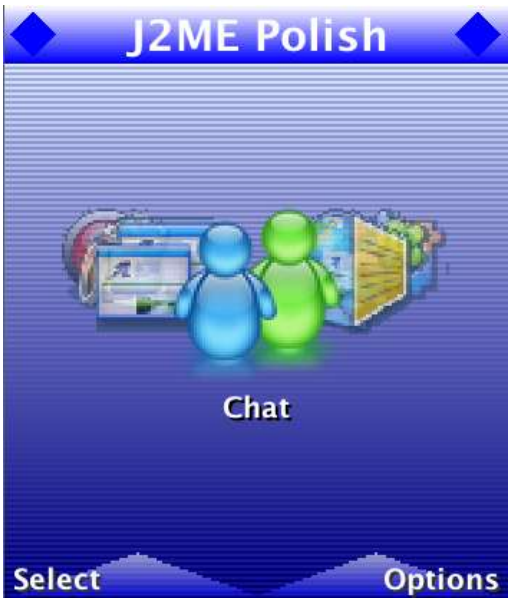

Figura 40: Interfaz realizada en J2ME utilizando J2ME Polish

Al igual que J2ME Polish, LWUIT y KUIX existen más alternativas a la API de objetos gráficos que ofrece J2ME por defecto. Cada una de ellas tiene sus características específicas y el uso de las mismas depende de los gustos que el desarrollador tenga y las capacidades que la aplicación requiera.

Para la futura mejora de este proyecto que sería la realización de una versión para dispositivo móvil, se podría utilizar esta tecnología, ya que entre sus ventajas está que es compatible con la mayoría de los dispositivos móviles (independientemente del sistema operativo) y su sencillez a la hora de crear interfaces, pues la creación es muy similar a la utilizada con las clases Swing y AWT ofrecidas por Java.

9.3.5.1.2. Interfaz en Android

Otras de las alternativas para realizar la interfaz para dispositivos móviles es la de realizarla para el sistema operativo Android. Este sistema operativo de la empresa Google, está basado en Linux para dispositivos móviles.

Las aplicaciones desarrolladas para Android son implementadas en código Java, que aunque no es exactamente el mismo que se utiliza en la plataforma Java original, si incluye la mayoría de las funciones del mismo. Así también comentar que se está desarrollando una versión de la tecnología Selenium para este sistema operativo, lo que haría más fácil la integración entre la interfaz y el código fuente.

Android ofrece el kit de desarrollo "Android SDK" que posee las librerías para la implementación de aplicaciones, incluyendo la que ofrece los componentes gráficos para la creación de interfaces.

Para la implementación de aplicaciones para Android existen plugins específicos para los principales entornos de desarrollo, como Netbeans o Eclipse. Estos plugins traen unos simuladores del sistema operativo Android, para ver cómo se ejecutaría la aplicación creada en un dispositivo móvil. Además ofrecen la posibilidad de implementar las interfaces gráficas de manera sencilla utilizando estas librerías. En la Figura 41 se puede ver un ejemplo del emulador de Android que incluye el plugin para Netbeans.

Figura 41: Plugin de Android para Netbeans

La futura mejora de este proyecto que sería la realización de una versión para el sistema operativo Android es una muy buena opción, ya que cada vez son más los dispositivos que utilizan este sistema operativo, además la API que incluye ofrece unos componentes gráficos muy avanzados, actuales y visualmente atractivos. En la Figura 42 se puede observar una interfaz realizada para Android.

Figura 42: Interfaz realizada para Android

A continuación se mostrará una tabla de equivalencias entre los elementos implementados en la interfaz creada con las clases SWING y AWT y sus correspondientes elementos realizando la interfaz con las librerías base de J2ME y con el SDK ofrecido por Android.

Estos elementos se corresponden con objetos creados de las clases que llevan su mismo nombre. Así por ejemplo una caja de texto editable por el usuario se creará utilizando un objeto de la clase “JFormattedTextField” si se implementa en SWING/AWT, un objeto de la clase “TextBox” si se implementa en J2ME o un objeto “EditText” si se implementa con el SDK de Android.

La intención de esta comparativa es ofrecer una garantía de la viabilidad de la re-implementación de la interfaz para dispositivos móviles de la aplicación utilizando estas tecnologías. Así se podrá obtener una correspondencia clara de por qué ele-

mentos es necesario sustituir los elementos implementados en la interfaz de la aplicación para obtener una versión para dispositivos móviles de la misma.

Elemento SWING/AWT	Elemento J2ME	Elemento Android SDK
jPanel: contenedor de elementos genérico.	Form	Absolute Layout
jFormattedTextField: caja de texto editable por el usuario.	TextBox	EditText
jCheckBox: selección múltiple de un conjunto de opciones.	List (Multiple) / ChoiceGroup	CheckBox
jRadioButton: selección exclusiva de un conjunto de opciones.	List (Exclusive) /ChoiceGroup	RadioButton
jLabel con texto: contenedor que contiene una cadena de texto para mostrar en la interfaz.	StringItem	TextView
jLabel con imagen: contenedor que contiene una imagen para mostrar en la interfaz.	ImageItem	ImageView

Tabla 57: Comparativa SWING/AWT, J2ME, Android SDK.

9.3.6. Detectar productos únicos

Una futura mejora muy importante sería el poder detectar los productos únicos, como ya hacen algunos Mashups. Esta funcionalidad sería la de saber a ciencia cierta que dos o más resultados son el mismo producto. El problema semántico que supone dificulta esta tarea, pues solo se podría saber comparando los nombres de los productos y este difiere en cada fuente utilizada.

Esta mejora posibilitaría la realización de múltiples opciones como el poder ofrecer al usuario una valoración o comentarios sobre el mismo producto como tam-

bién poder ofrecer una comparativa más amplia sobre el mismo producto ofertado en diferentes tiendas.

9.4. Conclusiones personales

Como conclusión personal sobre el desarrollo de la aplicación voy a comentar como ha sido el proceso de desarrollo de la misma desde mi propia experiencia.

9.4.1. Conclusiones del desarrollo

El proceso de desarrollo de la aplicación me ha resultado en general trabajoso. No tanto al principio como al final. El análisis y diseño de la aplicación fueron relativamente sencillos, ya que tras haber realizado un análisis de aplicaciones similares, los requisitos principales estaban bastante claros, así como también la elección de la arquitectura.

En cuanto a la implementación, tuve algunas dificultades en cada uno de los niveles así, voy a destacar las más relevantes:

- Respecto al Nivel de Usuario la implementación resultó bastante complicada ya que pese a ser una tecnología sencilla de manejar, no ofrece las posibilidades gráficas para que la interfaz quede bien conjuntada y visualmente atractiva, con lo cual el afán por querer dejarla lo más agradable posible a vista del usuario, tuvo como consecuencia muchos problemas y cambios en la misma.

- Considerando el Nivel de Integración, la complejidad residió en manejar eficientemente las peticiones que provienen de cada uno de los diferentes niveles. También el orden por relevancia resulto algo complejo, no tanto la implementación como el diseño del método utilizado para considerar que productos eran más relevantes que otros.

- En cuanto al Nivel de Fuente de Datos: la dificultad radicó en conseguir integrar la tecnología Selenium y el lenguaje XPath en un proyecto Java. Ya que de estas dos tecnologías mi desconocimiento era total y eso me dificulto las cosas en un principio al tener que documentarme

sobre ellas. Finalmente tras realizar una primera versión de la aplicación, me di cuenta que las expresiones XPath eran poco robustas con lo cual decidí sustituir estas expresiones por otras que la dotaran de mayor robustez.

Con la experiencia obtenida tras este desarrollo, si tuviera que empezarle de nuevo cambiaría algunas cosas como por ejemplo la tecnología a utilizar para realizar la interfaz, hubiera elegido una tecnología como HTML para realizar una versión Web desde el principio, ya que actualmente la mayoría de las nuevas aplicaciones están en formato Web por todas las ventajas de accesibilidad que ofrece este formato. También hubiera considerado el realizar una única clase genérica para los wrappers que sustituya a las siete clases existentes. Esta clase sería común para todos los wrappers y solo variarían las expresiones XPath de los elementos de cada uno de los portales.

Pese a todo he de decir que me ha gustado mucho este desarrollo, ya que durante el proceso he aprendido de muchas situaciones que tienen lugar en los desarrollo de los proyectos de desarrollo software, habiendo resultado una experiencia muy positiva para mí.

Anexo A: Manual de usuario

En este capítulo se va a ofrecer a los lectores el manual de usuario de la aplicación presentada en este documento. Este manual de usuario ha sido realizado con el objetivo de que los potenciales usuarios de la aplicación tengan acceso a una información sencilla y completa de cómo usar la aplicación, a la que puedan consultar antes del primer uso o en cualquier momento que tengan alguna duda sobre el mismo. Así, pues aunque la aplicación tiene un funcionamiento sencillo e intuitivo, he considerado positivo incluirlo, como complemento a este documento.

Se ha dividido este manual según las distintas funcionalidades que ofrece la aplicación, y estas funcionalidades vienen muy bien reflejadas en los casos de uso explicados en el punto 3.3 de este documento. Con lo cual los puntos a explicar en este manual serán:

- Búsqueda Genérica
- Búsqueda Específica
- Visualización de Resultados
- Ordenación de Resultados.

Comentar también, que se ha dotado al manual con imágenes de la aplicación para facilitar la comprensión del mismo. Sin más dilación, pasamos a ver las distintas funcionalidades de la aplicación incluidas en este manual.

Búsqueda genérica

Para realizar la búsqueda genérica, el usuario debe de estar situado en la pestaña “General”, esta viene representada en la Figura 43.

Figura 43: Búsqueda genérica

Una vez en la pestaña “General” se debe de hacer clic en la caja de texto central para que esta quede activada para poder introducir el texto. La caja de texto viene representada en la Figura 44.

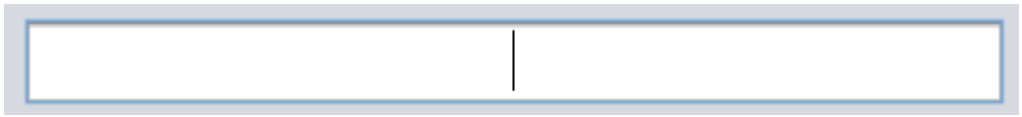

Figura 44: Caja de texto de búsqueda genérica

Una vez activada, introducimos el texto de búsqueda, es decir el texto que va a describir el producto o conjunto de productos que queremos buscar. Se recomienda ser lo más descriptivo y conciso posible, ofreciendo todos los datos que podamos del producto específico, así la búsqueda será más precisa.

Por ejemplo si queremos buscar un teléfono móvil iPhone con tecnología 3G , un buen texto de búsqueda sería el de la Figura 45.

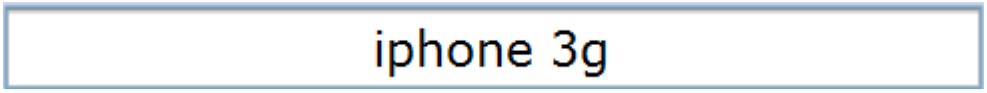

Figura 45: Inserción del texto de búsqueda genérica

Cuando ya hemos introducido el texto, se deberán seleccionar los portales donde queremos buscar. Esto se realizará en la sección de selección de portales, que viene representado en la Figura 46.

Figura 46: Selección de portales

En esta sección se irán seleccionando las casillas asociadas a cada uno de los portales contemplados. (Se recomienda no seleccionar más de 4 portales, para aumentar el rendimiento de la búsqueda).

Una vez terminada la selección, solo nos queda ejecutar la búsqueda. Esto se hará haciendo clic en el botón “Buscar” situado en la parte inferior de la interfaz. Este botón queda representado en la Figura 47.

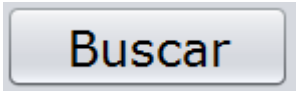

Figura 47: Botón de búsqueda

Una vez ejecutada la búsqueda, el navegador lanzado durante la inicialización de la aplicación, empezará a navegar por los portales webs seleccionados y realizará la búsqueda con el texto que hemos introducido. Tras esta acción, se mostrarán los resultados en la pestaña “Resultados”.

Búsqueda específica

Para realizar la búsqueda especifica el usuario debe de estar situado en cualquiera de las siguientes pestañas “Televisiones”, “Fotografía-MP3”, “Memoria”, o “Informática”, el procedimiento de búsqueda es el mismo para cada pestaña, lo cual solo explicaremos para una pestaña: “Memoria” que viene descrita en la Figura 48.

Figura 48: Interfaz de la pestaña "Memoria”

Una vez en la pestaña “Memoria”, seleccionamos uno de los productos contenidos en la misma, una vez seleccionado, se activarán los controles que muestran las características a elegir. Los controles correspondientes a la categoría de memorias USB se representan en la Figura 49.

Figura 49: Selección de características de la categoría "Memorias USB"

Una vez activados, seleccionamos las características que queremos de las distintas listas desplegables. Pudiendo dejar marcados las opciones por defecto: "Todas/Todos", que significa que buscará para todos los posibles valores de la lista, lo que le quita especificidad a la misma, pero le otorga una amplia variedad de resultados. En la Figura 50 se muestra como se selecciona un elemento de la lista desplegable, que elige el rango de precios de la memoria USB.

Figura 50: Características desplegadas en la categoría "Memorias USB"

Cuando ya hemos seleccionado las características, se deberá seleccionar los portales donde queremos buscar. Esto se realizará en la selección de tiendas que se muestra en la Figura 46.

En esta sección se irán seleccionando las casillas asociadas a cada uno de los portales contemplados. (Se recomienda no seleccionar más de 4 portales, para aumentar el rendimiento de la búsqueda).

Una vez terminada la selección, solo nos queda ejecutar la búsqueda. Esto se hará haciendo clic en el botón "Buscar" situado en la parte inferior de la interfaz. Este botón viene representado en la Figura 47.

Una vez ejecutada la búsqueda, el navegador lanzado durante la inicialización de la aplicación, empezará a navegar por los portales webs seleccionados y realizará una búsqueda con las opciones que hemos introducido en la búsqueda específica. Tras esta acción, se mostrarán los resultados en la pestaña “Resultados”.

Visualización de resultados

La visualización de resultados es un tarea automática, ya que tras realizar una búsqueda, la pestaña de resultados se activa automáticamente mostrando los resultados obtenidos en la búsqueda. La pestaña queda descrita en la Figura 51.

Figura 51: Pestaña “Resultados” de la aplicación

Los resultados son mostrados en esta pantalla por filas, cada fila está formada los siguientes atributos situados de izquierda a derecha en la interfaz:

- Una fotografía del producto.
- El logo del portal donde que oferta el producto.
- El nombre del producto.
- El precio en euros del producto.
- El precio en dólares del producto (si se precisara).

A su vez, las filas de resultados se agrupan en páginas, mostrando 20 filas por cada página. Con lo cual en esta interfaz se puede navegar por los resultados de dos formas distintas:

- **Navegación en la misma página:** Se puede navegar por los resultados en la misma página utilizando el scroll situado a la derecha de la interfaz. Así se pueden visualizar los 20 resultados por página.
- **Navegación entre páginas:** si los resultados obtenidos en la búsqueda son más de 20, se activarán los controles de navegación de páginas. Estos controles son los mostrados en la siguiente Figura 52.

Figura 52: Controles de navegación de resultados

Una vez que se pulse sobre uno de ellos, se navegará hacia la página siguiente o anterior dependiendo del botón pulsado, mostrándose los resultados en la interfaz.

Ordenación de resultados

La ordenación de resultados es una tarea que se realiza desde la pestaña "Resultados". Así, en la parte superior de esta pestaña se encuentran la lista desplegable (que se puede ver en la Figura 53) que nos permitirán ordenar los resultados en un orden o en otro. Estos órdenes son:

- Productos por relevancia.
- Productos por precio.
- Productos por portales.

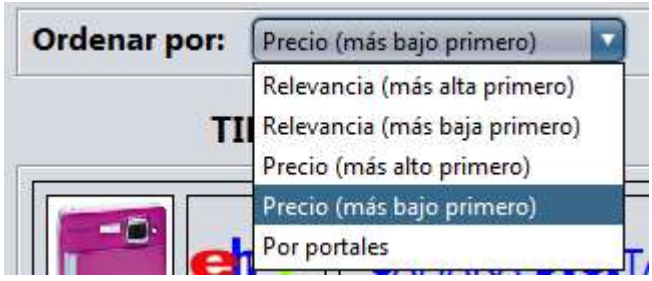

Figura 53: Controles de ordenación de resultados

Lo único que se necesita hacer para ordenar por uno de estos criterios, es seleccionar la opción requerida en la lista desplegable. En el momento en el que se seleccione una de ellas los productos se ordenarán por el criterio seleccionado, apareciendo así en la zona de muestra de resultados de la pestaña.

Opciones de la aplicación

La aplicación permite configurar una serie de parámetros para que el usuario pueda personalizarla como el desee. Así en la pestaña “Opciones” se definen una serie de propiedades modificables. Esta pestaña viene definida en la Figura 54.

Las opciones que se contemplan son las siguientes:

- **Número de productos máximos a obtener en un portal:** indica el número de productos que como máximo se van a ofrecer como resultado de la búsqueda en un determinado portal. Así para modificar este parámetro simplemente hay que hacer clic en la caja de texto y pulsar una nueva cifra.

  La suma de todos los productos no puede ser más de 100, con lo cual se mostrará un mensaje de error en ese caso indicando el motivo. Por defecto está puesto a 10 productos a mostrar por portal.

- **Tiempo límite de búsqueda en cada portal:** se define el tiempo en segundos que como máximo puede estar la aplicación realizando la búsqueda en un portal, sin que muestre un error de conexión. Este valor es parametrizable para que el usuario pueda ajustarlo dependiendo de su conexión a Internet. Así para modificar este parámetro simplemente hay que hacer clic en la caja de texto y pulsar una nueva cifra. Por defecto está puesto a 10000 segundos.

- **No mostrar imágenes:** haciendo clic en esta casilla, se activará la opción de que no se muestren las imágenes de los productos en la interfaz de “Resultados”, esto incrementa el rendimiento al no tener que realizar una carga de estas imágenes.

- **No realizar conversión de monedas:** activando esta casilla, los precios de los productos no serán convertidos de dólares a euros.

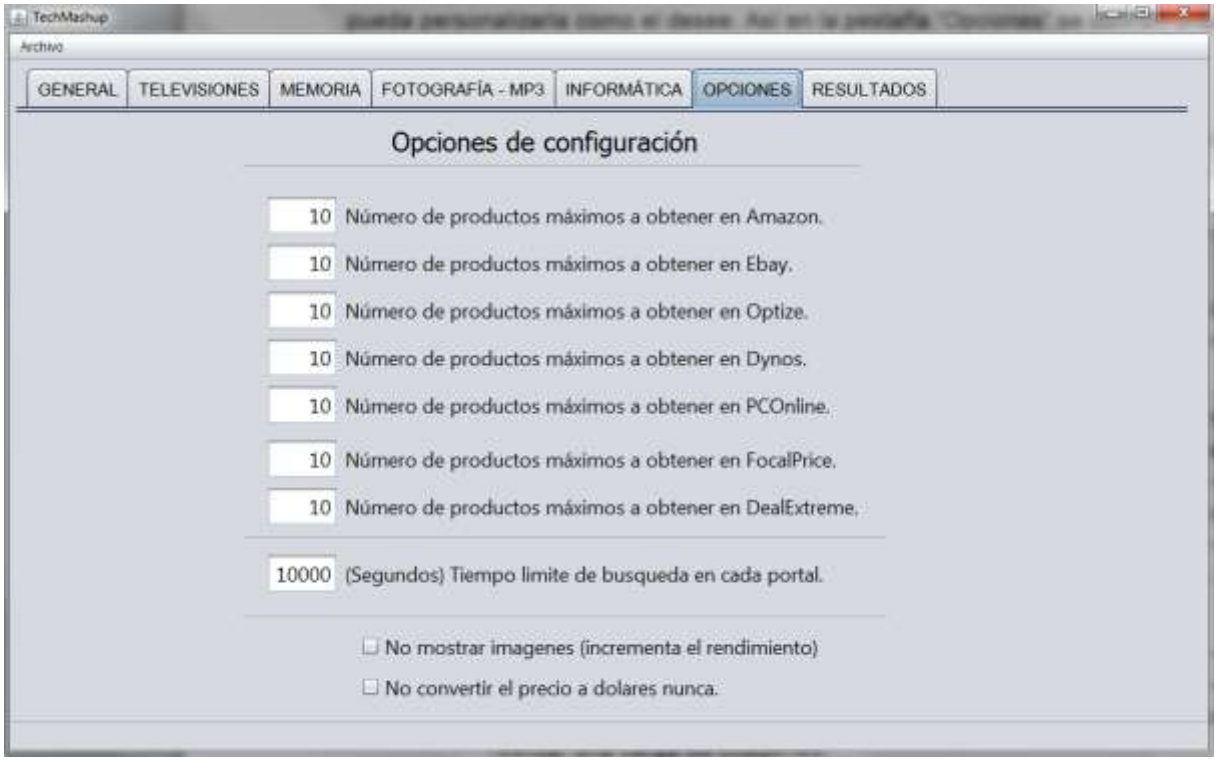

Figura 54: Opciones de configuración

Anexo B: Código fuente de la aplicación

En este anexo se mostrará el código fuente más relevante de la aplicación. Se irán exponiendo las partes que se creen más importantes en cuanto a la implementación del proyecto. Estas partes son: la implementación de la entidad mediador, cuyo interés reside en ver como se ha gestionado las comunicaciones en la aplicación, también veremos cómo se han implementado la ordenación de productos y finalmente se mostrará la implementación de una entidad wrapper, donde la importancia reside en observar el uso de la tecnología Selenium y XPath. Este anexo se organizará en entidades y dentro de esta en funcionalidades.

Código fuente de la entidad Mediador

Llamada a los wrappers.

La Figura 55, muestra la primera parte representativa del código del método principal de la entidad Mediador: el método “getProducts()”, cuya función es recibir las opciones de búsqueda del Nivel de Usuario y dependiendo de ellas ir llamando a los diferentes wrappers seleccionados para obtener los resultados de sus búsquedas y así poder devolverlos al nivel superior.

```
public static List<Product> getProducts(String searchText, boolean focal, boolean extreme, boolean amazon, boolean ebay,
boolean optize, boolean dynos, boolean pcCity, boolean pcOnline, String textForFocal, String textForExtreme, String textForAmazon,
String textForOptize, String textForEbay, String textForCity, String textForDynos, String textForOnline, double initialLimitPrice,
double finalLimitPrice, Boolean matchWordFilter, Boolean shopsFilter, Boolean imageFilter) throws Exception {

    System.out.println("[MEDIATOR GET_PRODUCTS STARTS]");
    if (searchText.equals("")) {
        System.out.println("Debes introducir un texto para la busqueda.");
    } else {
        list.clear();
        list2.clear();
        list3.clear();

        if (focal) {
            if (textForFocal != null) {
                focalProducts = FocalPriceWrapper.getInfo(textForFocal);
            } else {
                focalProducts = FocalPriceWrapper.getInfo(searchText);
            }
            for (int i = 0; i < focalProducts.length; i++) {
                if (focalProducts[i].getName() != null) {
                    list.add(focalProducts[i]);
                }
            }
        }
```

Figura 55: Primera parte del método getProducts()

Como se puede observar en el código, el método recibe como parámetro las distintas opciones que el usuario ha seleccionado en la interfaz de la aplicación. Estos parámetros son:

- **Texto de búsqueda genérica:** texto que el usuario ha introducido manualmente para la realización de la búsqueda genérica.

- **Selectores de portales:** un booleano para cada portal, que tomará valor "true" si el portal ha sido seleccionado para la búsqueda por el usuario, o valor "false" en caso contrario.

- **Textos de búsqueda específica:** en el caso de la búsqueda específica se enviará un texto especializado a cada portal seleccionado por el usuario.

- **Límites en el precio:** en el caso de la búsqueda específica el usuario puede seleccionar un rango de precios en el que quiere que se encuentren los productos devueltos como resultado. Así se enviarán tanto el límite inferior como el superior.

- **Opciones de ordenación:** distintas opciones de filtrado u orden de los productos, como por ejemplo no mostrar la imagen de los productos.

También podemos observar cómo se comparan los booleanos referentes a la selección de cada portal y en el caso de que estén activados (con valor a "true"), se llama al wrapper asociado (a través del método getInfo()) , donde se realizará la búsqueda para más tarde almacenar los resultados devueltos en una lista de productos.

Finalmente como podemos ver en la Figura 56, que representa el final del método getProducts(), se establecen los links de las imágenes de los portales de todos los productos añadidos a la lista de productos del resultado, esto se realiza por medio del método setShopImageLink() . También se ajustan los precios a un formato único, ya que no todos los portales lo representan de la misma forma, en algunos se utilizan puntos para los decimales y en otros comas. Esta última función de realiza mediante la llamada al método adjustingPriceFormat(). El código de estos dos últimos métodos no será mostrado ya que no se considera de especial relevancia para ello. Finalmente la lista de productos obtenida en las búsquedas será devuelta al Nivel de Usuario.

```
        list = setShopImageLink(list);

        list = adjustingPriceFormat(list);


        return list;
    }
```

Figura 56: Segunda parte del método getProducts()

Código fuente de la entidad Filter

Orden de relevancia

En la Figura 57 se muestra el inicio del código del método “wordsMatchFilter()” de la entidad Filter, cuya función es ordenar los productos por relevancia una vez lo ha requerido el usuario. Estos productos son recibidos de la entidad Mediador y devueltos a la misma ordenados, para que se transfieran al Nivel de Usuario.

El funcionamiento de este método es comparar el nombre de cada producto con el texto de búsqueda e ir contabilizando las palabras en las que coinciden. Así cuantas más coincidencias tengan ambos textos, mayor será la relevancia del producto.

```
public static List<Product> wordsMatchFilter(List<Product> list, String searchText) throws Exception {
   System.out.println("[WORDS_MATCH_FILTER] STARTS");
    iterator4 = list.iterator();
    // Calculating the count of equal words
    while (iterator4.hasNext()) {
        int count;
        count = 0;
        Product aux = (Product) iterator4.next();
        aux.setCount(0);

        /*Cleaning names removing special characters*/
        nameAux = aux.name;
        nameAux = nameAux.replace('(', ' ');
        nameAux = nameAux.replace(')', ' ');
        nameAux = nameAux.replace(',', ' ');
        //nameAux = nameAux.replace('.', ' ');
        nameAux = nameAux.replace(':', ' ');
        nameAux = nameAux.replace('_', ' ');
        nameAux = nameAux.replace('-', ' ');
        nameAux = nameAux.replace('+', ' ');
        nameAux = nameAux.replace('"', ' ');
        nameAux = nameAux.toLowerCase();
        searchText = searchText.toLowerCase();

        /*Array with every word of searchText*/
        searchTextAuxArray = searchText.split(" ");

        /*Array with every word of special name*/
        nameAuxArray = nameAux.split(" ");
```

Figura 57: Primera parte del método wordsMatchFilter()

Como se puede observar en la figura, el método recibe dos parámetros que explicamos a continuación:

- **Lista de productos:** lista de productos obtenida como resultado de las búsquedas.
- **Texto de búsqueda:** texto introducido por el usuario que se utilizará para las comparaciones con el nombre de los productos.

En la primera parte de este método podemos ver cómo se va recorriendo la lista y procesando cada producto. Lo primero que se realiza con un producto es eliminar todos los caracteres de su nombre, que no sean letras. Esto se realiza para evitar problemas en la comparación semántica de palabras con el texto de búsqueda. Finalmente se separan por palabras tanto el texto de búsqueda como el nombre del producto, guardándolas en un array que contendrá cada una de las palabras que forman cada cadena de texto.

En la Figura 58, se observa la segunda parte de este método. En ella vemos que una vez separados los dos textos por palabras se van comparando una a una semánticamente y en el caso de coincidencia se aumenta un contador reservado para este fin.

Una vez obtenido el valor del contador para cada producto, se realiza una llamada al método genérico “sort()”, de la clase Collections de Java, que ordenará la lista de mayor a menor número de contador. Finalmente cuando obtenemos la lista ordenada, se devuelve de nuevo a la entidad Mediador.

```
        /* Comparing every word of the name with every word of the searchText*/
        for (int j = 0; j < searchTextAuxArray.length; j++) {
            wordToCompare = searchTextAuxArray[j];
            equalWord = false;
            for (int i = 0; i < nameAuxArray.length; i++) {
                if (wordToCompare.equals(nameAuxArray[i])) {
                    equalWord = true;
                }
            }
            if (equalWord == true) {
                // If the words match count ++;
                count++;
            }
        }
        aux.setCount(count);
    }

    //Sorting the list by count
    Collections.sort(list, new Product());
     System.out.println("[WORDS_MATCH_FILTER] ENDS");
    return list;
}
```

Figura 58: Segunda parte del método wordsMatchFilter()

Rango de precios

La Figura 59 muestra el código del método de la entidad Filter: “priceFilter()”, cuya función es seleccionar los productos que se encuentran en el rango de precios introducido por el usuario. Estos productos son recibidos de la entidad Mediador y devueltos a la misma una vez realizado el filtrado de productos, para que sean transferidos al Nivel de Usuario.

Así, este método recibe tres parámetros, estos son:

- **Lista de productos:** lista de productos con los productos a filtrar.
- **Límite inferior:** límite inferior del rango de precios elegido por el usuario.
- **Límite superior:** límite superior del rango de precios elegido por el usuario.

El funcionamiento de este método es bastante sencillo. Se trata de ir recorriendo la lista producto a producto y comprobando que el precio del producto este dentro de los límites establecidos, añadiendo a una nueva lista los productos que cumplen esta características. Esta lista será devuelta a la entidad Mediador.

```
public static List<Product> priceFilter(List<Product> listToFilter, double initialLimitPrice, double finalLimitPrice ) {
    System.out.println("[PRICE_FILTER] STARTS");
     List<Product> newList3 = new ArrayList<Product>();

            iterator5 = listToFilter.iterator();

            /*PRICES FILTER*/
            while (iterator5.hasNext()) {
                Product aux = (Product) iterator5.next();
                if (aux.price >= initialLimitPrice && aux.price <= finalLimitPrice) {
                    newList3.add(aux);
                } else {
                }

            }
    System.out.println("[PRICE_FILTER] ENDS");
     return  newList3;
}
```

Figura 59: Método priceFilter()

Orden de precios

El orden de los productos por precios (tanto ascendente como descendente) se ha realizado implementando un método que ordena la lista dependiendo del criterio que se le pase por parámetro. Este método es “sortProducts()” y se puede ver en la Figura 60. Recibe como parámetros:

- **Lista de productos**: lista de productos a ordenar por precio.
- **Propiedad:** propiedad de los elementos de la lista por la que se ordenara la misma.

Para la implementación de esta función nos hemos apoyado en él método “sort()” de la clase Collections de Java (que hemos comentado en la explicación del método wordsMatchFilter()) , así como de la clase “Comparable” de Java, que ofrece unas propiedades a sus objetos para que estos se puedan comparar de una forma sencilla utilizando el método compareTo() de la misma.

Este método ordena los productos de mayor a menor precio (orden descendente). Así para obtener el orden inverso, solo tenemos que invertir la lista devuelta por este método.

```
public static void sortProducts(List listToFilter, final String propiedad) {
    Collections.sort(listToFilter, new Comparator() {

        public int compare(Object obj1, Object obj2) {

            Class clase = obj1.getClass();
            String getter = "get" + Character.toUpperCase(propiedad.charAt(0)) + propiedad.substring(1);
            try {
                Method getPropiedad = clase.getMethod(getter);

                Object propiedad1 = getPropiedad.invoke(obj1);
                Object propiedad2 = getPropiedad.invoke(obj2);

                if (propiedad1 instanceof Comparable && propiedad2 instanceof Comparable) {
                    Comparable prop1 = (Comparable) propiedad1;
                    Comparable prop2 = (Comparable) propiedad2;
                    return prop1.compareTo(prop2);
                }//CASO DE QUE NO SEA COMPARABLE
                else {
                    if (propiedad1.equals(propiedad2)) {
                        return 0;
                    } else {
                        return 1;
                    }

                }

            } catch (Exception e) {
                e.printStackTrace();
            }
            return 0;
        }
    });
```

Figura 60: Método sortProducts()

Código fuente de la entidad View

Instanciación del Navegador

Antes de proceder a explicar la funcionalidad de los wrappers, es necesario comentar que en la clase de inicio de nuestra aplicación “View” se han declarado los objetos necesarios para la utilización de la tecnología Selenium. Estos objetos se han declarado en esta clase, para poder levantar una instancia del navegador al iniciar la aplicación. En la Figura 61 se puede ver esta declaración:

```
static public WebDriver driver = new FirefoxDriver();

static public Selenium selenium = new WebDriverBackedSelenium(driver, "http://www.google.es/");
```

Figura 61: Instanciación del navegador

Como se observa tenemos un objeto WebDriver (clase genérica que implementa los manejadores de todos los navegadores soportados por Selenium) que creamos como una instancia de FirefoxDriver (clase que representa el manejador especifico de Selenium para el navegador Mozilla Firefox).

Finalmente creamos un objeto de tipo Selenium (clase que nos dará acceso a todos los métodos que implementan las distintas funcionalidades que ofrece esta tecnología) al que le pasamos en su inicialización el manejador para Firefox creado anteriormente y una URL que servirá como página de inicio al lanzarse el navegador.

Así el navegador se lanzará desde esta clase de inicio, mediante la función open() que ejecutará el navegador Mozilla Firefox e introducirá la URL pasada en la declaración del objeto Selenium. Esta llamada se puede ver en la siguiente figura:

```
selenium.open();
```

Figura 62: LLamada open()

Código fuente de la entidad Wrapper

Búsqueda en el portal

Debido a que todos los wrappers dedicados a portales de venta implementados tienen la misma estructura, solo mostraremos el código de uno de ellos. En la figura 63 muestra el inicio del código del método getInfo() de la entidad OptizeWrap-

per. Cuya función es navegar hacia la página del portal asociado, realizar la búsqueda con el texto que proviene de la entidad Mediador, extraer la información de los productos, almacenarla y devolverla al nivel superior.

```
public static Product[] getInfo(String searchText) throws Exception {
        try{

        if (searchText.contains("laptop")) {
            searchText = searchText.replace("laptop", "RAM");
            View.driver.get("http://www.optize.es/servlet/ORDENADORES_PORTATILES_16407_catOp

        } else if (searchText.contains("desktop")) {
            View.driver.get("http://www.optize.es/servlet/navigation?category=16406&pag=1&or
            searchText = searchText.replace("desktop", "RAM");
        } else if (searchText.contains("Pantalla")) {

            View.driver.get("http://www.optize.es/servlet/MONITORES_16449_catOptize.htm");
```

Figura 63: Primera parte del método getInfo()

En esta primera parte del método se puede observar que solo recibe un parámetro, este es el texto de búsqueda a insertar en el buscador del portal asociado.

Lo primero que se realiza es comprobar si la cadena de texto recibida contiene alguna de las palabras que se han generado en el Nivel de Integración, en este caso estaríamos en la búsqueda específica y utilizando la tecnología Selenium, navegaríamos a la sección específica del portal Optize del tipo de producto que se quiere buscar. Estas palabras generadas son específicas para cada portal, por ejemplo, para Optize se utiliza la palabra “Pantalla” para referirnos a los monitores de PC, con lo cual si el texto contiene esta palabra, se navegará a la sección específica de Optize para monitores de PC. Para esta navegación utilizamos la función “get()” del objeto asociado al navegador lanzado en el inicio de la aplicación. Esta función navega hacia la URL que recibe como parámetro.

Una vez que hemos accedido a la página del portal, ya sea la sección de la categoría del producto (en el caso de la búsqueda específica) o a la página inicial (en el caso de la búsqueda genérica), procedemos a realizar la búsqueda con el texto recibido. Para ello utilizaremos los métodos de Selenium: type() y click() , que podemos observar en La Figura 64.

```
View.selenium.type("searchText", searchText);
View.selenium.click("//input[@type='image']");
```

Figura 64: LLamadas a type() y click()

Estos métodos realizan la siguiente funcionalidad:

- **type (locator, value):** escribe en el elemento HTML definido por “locator”, la cadena de texto que contiene “value”.
- **click(locator):** realiza un clic con el ratón en el elemento seleccionado.

Estas funciones se utilizan para escribir el texto en la caja de búsqueda de la página web y hacer clic en el botón que ejecuta dicha búsqueda.

Una vez que se ha cargado la página de resultados, se lleva a cabo la extracción de los datos de los productos obtenidos, así se tratará cada característica que queramos obtener de forma separada.

En la Figura 65 se puede observar el código correspondiente a la extracción del nombre y del link del producto.

```
productsName = new String[numberResults];
productsLinks = new String[numberResults];
j = 0;
for (int i = 1; i < numberResults; i++) {
    consultBeg = "xpath=/html/body/table/tbody/tr[2]/td[3]/table[2]/tbody/tr[";
    numberName = String.valueOf(i);
    consultEnd = "]/td[4]/small/a@title";
    consult = consultBeg + numberName + consultEnd;
    try {
        productsName[j] = View.selenium.getAttribute(consult);
        optizeProducts[j].setName(productsName[j]);
        consultEnd = "]/td[4]/small/a@href";
        consult = consultBeg + numberName + consultEnd;
        productsLinks[j] = "http://www.optize.es/servlet/" + View.selenium.getAttribute(consult);
        optizeProducts[j].setLink(productsLinks[j]);
        optizeProducts[j].setShop("Optize.es");
        j++;
    } catch (SeleniumException e) {
    }

}
```

Figura 65: Segunda parte del método getInfo()

Lo primero que se realiza es la declaración de las variables donde se guardarán estos atributos del producto. Después vamos recorriendo todos los resultados obtenidos y para cada uno de ellos realizamos los siguientes pasos para la extracción de sus datos:

- **Creamos la consulta XPath:** se crea en una variable la ruta XPath y se va actualizando para cada resultado obtenido.
- **Obtenemos el atributo:** se extrae la información mediante la utilización de la función ofrecida por Selenium “getAttribute()”. Esta función obtiene el atributo especificado en la ruta XPath pasada como argumento. En este caso los atributos “title” y”href” que se corresponden con el nombre y el link del producto respectivamente.

- **Guardamos los atributos obtenidos:** cada atributo obtenido de cada producto se guardan en un array de objetos de la estructura unificada de la clase Product.

Estos pasos se realizan para la obtención de todos los atributos de un producto, así además del nombre y el link, se hará para el precio y la imagen del mismo. Finalmente estos productos son devueltos a la entidad Mediador.

Glosario de términos y bibliografía

Glosario de términos

1. **3G:** abreviación de tercera-generación de transmisión de voz y datos a través de telefonía móvil
2. **Android:** es un sistema operativo basado en Linux para dispositivos móviles, tales como teléfonos inteligentes o tablets.
3. **API:** (Application Programming Interface) es el conjunto de funciones y procedimientos que ofrece una biblioteca para ser utilizado por otro software cómo una capa de abstracción.
4. **Applet:** componente de una aplicación que se ejecuta en el contexto de otro programa, por ejemplo un navegador web.
5. **Array:** es una zona de almacenamiento continuo en un programa, que contiene una serie de elementos del mismo tipo.
6. **AWT:** (Abstract Window Toolkit) es un kit de herramientas de gráficos, interfaz de usuario, y sistema de ventanas independiente de la plataforma original de Java.
7. **Check box:** es un elemento de la interfaz gráfica de usuario que permite al usuario hacer selecciones múltiples de un conjunto de opciones.
8. **Framework:** estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, con base en la cual otro proyecto de software puede ser organizado y desarrollado.
9. **Google Finance:** sitio web de Google que muestra datos económicos sobre los diferentes mercados internacionales, así como noticias sobre economía o datos en tiempo real sobre el cambio de divisas.
10. **HTML:** (HyperText Markup Language), es el lenguaje de marcado predominante para la elaboración de páginas web.
11. **IDE:** (Integrated Development Environment) es un programa informático compuesto por un conjunto de herramienta de programación.
12. **iPhone:** teléfono inteligente multimedia con conexión a Internet, pantalla táctil capacitiva (con soporte multi-táctil) y una interfaz de software minimalista de la compañía Apple Inc.
13. **J2ME:** (Java 2 Micro Edition) es una especificación de un subconjunto de la plataforma Java orientada a proveer una colección certificada de APIs de desarrollo de software para dispositivos móviles.

14. **JavaScript:** es un lenguaje de programación interpretado. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.
15. **Mashup:** una página web o aplicación que usa y combina datos, presentaciones y funcionalidad procedentes de una o más fuentes para crear nuevos servicios.
16. **MP3:** formato muy utilizado de compresión de audio digital patentado que usa un algoritmo con pérdida para conseguir un menor tamaño de archivo.
17. **MP4:** es un formato contenedor que se utiliza para almacenar los formatos audiovisuales especificados por ISO/IEC y el grupo MPEG.
18. **PDA:** (Personal Digital Assistant) es un ordenador de mano originalmente diseñado como agenda electrónica con un sistema de reconocimiento de escritura.
19. **Plugin:** Complemento o aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica.
20. **Radio button:** es un tipo de elemento de interfaz gráfica de usuario que permite al usuario elegir una de un conjunto predefinido de opciones.
21. **Scroll**: objeto de una interfaz gráfica que permite visualizar las partes de la misma que quedan ocultas y que no se ajustan al espacio definido en la pantalla para albergarla.
22. **SDK:** (Software Development Kit) conjunto de herramientas de desarrollo que permite a un programador crear aplicaciones para un sistema concreto.
23. **SWING:** biblioteca gráfica para Java. Incluye elementos para la interfaz gráfica de usuario como cajas de texto, botones, listas desplegables y tablas.
24. **UML:** (Unified Modeling Language) es el lenguaje de modelado de sistemas de software más utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema.
25. **URL:** (Uniform Resource Locator), es una secuencia de caracteres, de acuerdo a un formato modélico y estándar, que se usa para nombrar recursos en Internet para su localización o identificación.
26. **XML:**(eXtensible Markup Language), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C).
27. **XPath:** un lenguaje de consulta que permite construir expresiones que recorren y procesan un documento XML.

Bibliografía

1. Página oficial de Selenium: http://seleniumhq.org/
2. Página oficial de Wikipedia: http://es.wikipedia.org
3. Página oficial de Netbeans: http://netbeans.org/
4. Página oficial de Eclipse: http://www.eclipse.org/
5. Página oficial de Amazon: http://www.amazon.com/
6. Página oficial de DealExtreme: http://www.dealextreme.com/
7. Página oficial de FocalPrice: http://www.focalprice.com/
8. Página oficial de Ebay: http://www.ebay.es
9. Página oficial de Optize: http://www.optize.es/
10. Página oficial de Dynos: http://www.dynos.es/
11. Página oficial de PC-Online: http://www.pc-online.net/
12. Página oficial de Q-Compare: http://q-compare.com/
13. Página oficial de Flattvpice.com: http://www.flattvprice.com/
14. Página oficial de GotFreeShipping: http://gotfreeshipping.com/
15. Página oficial de Kalmeo: http://www.kalmeo.org/
16. Página oficial de Lwuit: http://lwuit.java.net/
17. Página oficial de J2ME Polish: http://www.enough.de/products/j2me-polish/
18. W3Schools: http://www.w3schools.com/